问题描述+解决方案+效果截图

612个Web前端开发实用案例

HTML5+CSS3+jQuery+jQuery UI+SVG

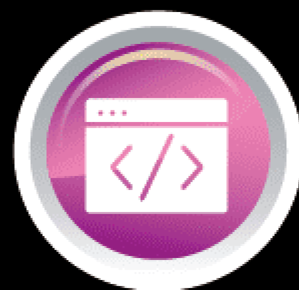
HTML5+CSS3 炫酷应用实例集锦



罗帅 罗斌 汪明云 编著







清华大学出版社

HTML5＋CSS3 炫酷应用实例集锦

罗　帅　罗　斌　汪明云　编著

清华大学出版社
北　京

内 容 简 介

本书采用问题描述＋解决方案的模式，以 HTML5、CSS3、jQuery、jQuery UI、SVG 等新技术为核心，列举了 600 多个实用性极强的 Web 前端开发技术，旨在帮助广大读者快速解决实际开发过程中面临的诸多问题，从而提高项目的开发效率，拓展应用领域。全书内容分为文字、图像、动画、视频、元素、布局、选择器、存储、其他 9 部分，以所见即所得、所学即所用的速成思维展示了过渡动画、关键帧动画、滤镜、选择器、计数器、伪元素、盒子、沙箱、画布等 Web 前端技术的具体应用，揭秘了百度地图定位、响应式页面布局、散列图片布局、瀑布流图片布局、旋转圆弧滑出菜单、批量插入与自动编号、盒子模型、图像与文字特效、多饼图绘制等诸多炫酷创意实例的实现过程。

为了突出实用性和简洁性，本书在演示或描述这些实例时力求针对性地解决问题，并且所有实例均配有插图。本书适合作为 Web 前端开发人员的案头参考书，无论是对于编程初学者还是编程高手，本书都极具参考和收藏价值。

图书在版编目（CIP）数据

HTML5＋CSS3 炫酷应用实例集锦/罗帅，罗斌，汪明云编著. —北京：清华大学出版社，2018
ISBN 978-7-302-49918-3

Ⅰ. ①H…　Ⅱ. ①罗… ②罗… ③汪…　Ⅲ. ①超文本标记语言－程序设计 ②网页制作工具
Ⅳ. ①TP312.8 ②TP393.092.2

中国版本图书馆 CIP 数据核字(2018)第 055448 号

责任编辑：黄　芝　王冰飞
封面设计：刘　键
责任校对：胡伟民
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：210mm×285mm　　**印　　张：**47　　**字　　数：**1325 千字
版　　次：2018 年 9 月第 1 版　　**印　　次：**2018 年 9 月第 1 次印刷
印　　数：1～1500
定　　价：149.00 元

产品编号：077383-01

前言

foreword

2014年10月28日，W3C的HTML工作组发布了HTML5的正式推荐标准；2015年4月9日，W3C的CSS工作组发布了CSS基本用户接口模块(CSS Basic User Interface Module Level 3，CSS3 UI)的标准工作草案。实际上，HTML5和CSS3都是渐进的标准，HTML5的第一份正式草案于2008年1月22日公布，从此Web前端就出现了HTML5的多种新技术，并且许多主流浏览器相继跟进支持，呈现了HTML5和CSS3的炫酷功能。

W3C CEO Jeff Jaffe博士表示：HTML5将推动Web进入新的时代。不久以前，通过Web还只是看一些基础文档，而如今Web是一个极其丰富的平台，通过Web可以做很多事情。

CSS(Cascading Style Sheet，层叠样式表)用于对页面的布局、字体、颜色、背景和其他效果的实现进行更加精确的控制。CSS3是CSS技术的升级版本，CSS3开发是朝着模块化方向发展的，这些模块包括盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏布局等。本书未根据HTML5、CSS3、SVG的技术类别进行篇章的划分，而是将它们融合在一起，按照实例功能进行篇章的划分。实际上，各种技术都是为了实现某一目标，你中有我，我中有你，难分难解。

本书采用问题描述+解决方案的模式，以HTML5、CSS3、jQuery、jQuery UI、SVG等新技术为核心，列举了600多个实用性极强的Web前端开发技术，旨在帮助广大读者快速解决实际开发过程中面临的诸多问题，从而提高项目的开发效率、拓展应用领域。全书内容分为文字、图像、动画、视频、元素、布局、选择器、存储、其他9部分，以所见即所得、所学即所用的速成思维展示了过渡动画、关键帧动画、滤镜、选择器、计数器、伪元素、盒子、沙箱、画布等Web前端技术的具体应用，揭秘了百度地图定位、响应式页面布局、散列图片布局、瀑布流图片布局、旋转圆弧滑出菜单、批量插入与自动编号、盒子模型、图像与文字特效、多饼图绘制等诸多炫酷创意实例的实现过程。

本书实例在Windows 10操作系统下使用IntelliJ IDEA 14.0.2开发工具编写，测试浏览器主要采用Google Chrome浏览器，如书中无特别说明，请在以上环境中学习本书的实例。如果在Google Chrome浏览器中无法正常测试，请用火狐浏览器或IE浏览器的较新版本进行测试。实际上，本书中的大多数实例均可单文件测试，即在源代码文件夹中右击HTML源文件，例如myHtmlA058.html，在右键菜单中选择“打开方式(H)”→Google Chrome命令，即可在谷歌浏览器中正常运行该实例。在记事本程序中可查看源代码。

全书的所有内容和思想并非一人之力所能及，而是凝聚了众多热心人士的智慧并经过充分的提炼和总结而成，在此对他们表示崇高的敬意和衷心的感谢！本书编写人员包括罗帅、罗斌、汪明云、曹勇、陈宁、邓承惠、邓小渝、范刚强、何守碧、洪亮、洪沛林、江素芳、蓝洋、雷国忠、雷惠、雷玲、雷平、雷治英、刘恭德、刘兴红、罗聃、唐静、唐兴忠、童缙嘉、汪兰、王彬、王伯芳、王年素、王正建、吴多、吴诗华、杨开平、杨琴、易伶、张志红、郑少文等，终稿由罗斌统筹完成。由于时间关系和作者水平原因，本书内容

有可能存在少量对 HTML5、CSS3、SVG 等技术认识不全面或偏颇的地方，以及一些疏漏和不当之处，敬请读者批评指正。

读者可将购书凭证发送至邮箱 huangzh@tup.tsinghua.edu.cn 索取本书源代码。

罗 帅 罗 斌

2018 年 5 月于重庆渝北

目录

Contents

第1部分 文字

001 在画布上创建向上的3D拉影文字

此实例主要通过设置阴影属性值实现在画布上创建向上3D拉影的特效文字。当在浏览器中显示该页面时，单击“在画布上创建向上3D拉影的特效文字”按钮，将在下面显示向上3D拉影的特效文字“炫酷实例集锦”，如图001-1所示。有关此实例的主要代码如下。

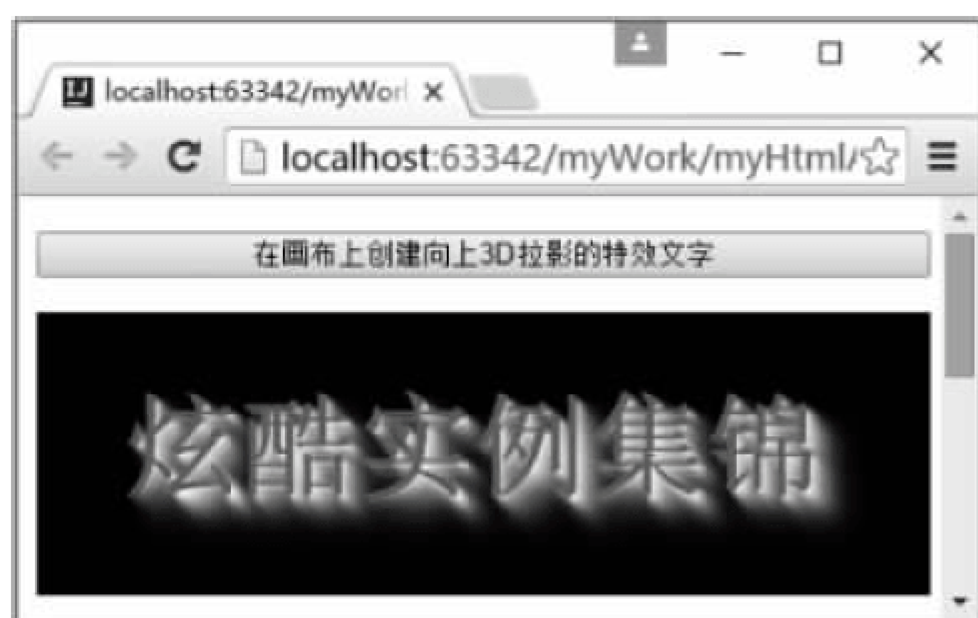


图 001-1

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {     //在画布上创建向上3D拉影的特效文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    myContext.font = '40pt 宋体';
    var myHeight = 0;
    myContext.fillStyle = "black";
    myContext.fillRect(0, myHeight, 410, 130);
    for (var i = 0; i < 10; i++) {
     myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
     //特效文字一
     myContext.shadowOffsetX = i * 2;
     myContext.shadowOffsetY = i * 2;
     //特效文字二
     //myContext.shadowOffsetX = - i * 2;
```

```
        //myContext.shadowOffsetY = - i * 2;
        myContext.shadowBlur = i * 2;
        myContext.fillStyle = "rgba(127, 127, 127, 1)";
        myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
  } });});
 </script></head>
<body><p><input type = "button" value = "在画布上创建向上 3D 拉影的特效文字" id = "myBtn" style =
"width:410px"></p>
<canvas id = "myCanvas" width = "410" height = "130"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，shadowColor 表示设置画布的阴影颜色；shadowOffsetX 表示阴影相对于文字的水平距离，0 表示两者水平位置重合；shadowOffsetY 表示阴影相对于文字的垂直距离，0 表示两者垂直位置重合；shadowBlur 表示阴影模糊效果，值越大模糊越厉害。fillText()方法用于在画布上绘制填色的文字，文字的默认颜色是黑色，该方法的语法声明如下。

```
fillText(text,x,y,maxWidth);
```

其中，参数 text 用于规定在画布上输出的文字；参数 x 表示开始绘制文字的 x 坐标位置（相对于画布）；参数 y 表示开始绘制文字的 y 坐标位置（相对于画布）；参数 maxWidth 是可选参数，表示允许的最大文字宽度，以像素计算。

此实例的源文件名是 myHtmlA058.html。

002　在画布上创建向下的 3D 拉影文字

此实例主要通过设置阴影属性值实现在画布上创建向下 3D 拉影的特效文字。当在浏览器中显示该页面时，单击“在画布上创建向下 3D 拉影的特效文字”按钮将在下面显示向下 3D 拉影的特效文字“炫酷实例集锦”，如图 002-1 所示。有关此实例的主要代码如下。



图　002-1

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    $("#myBtn").click(function() {      //在画布上创建向下 3D 拉影的特效文字
     var myCanvas = document.getElementById("myCanvas");
     var myContext = myCanvas.getContext('2d');
     myContext.font = '40pt 宋体';
```

```
        var myHeight = 0;
        myContext.fillStyle = "black";
        myContext.fillRect(0, myHeight, 410, 130);
        for (var i = 0; i < 10; i++) {
         myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
         myContext.shadowOffsetX = 0;
         myContext.shadowOffsetY = - i * 2;
         myContext.shadowBlur = i * 2;
         myContext.fillStyle = "rgba(127, 127, 127, 1)";
         myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
   } });});
</script></head>
<body><p><input type = "button" value = "在画布上创建向下 3D 拉影的特效文字" id = "myBtn" style =
"width:410px"></p>
<canvas id = "myCanvas" width = "410" height = "120"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，shadowColor 表示设置画布的阴影颜色；shadowOffsetY 表示阴影相对于文字的垂直距离，0 表示两者垂直位置重合，当 shadowOffsetY 值为正数时产生的阴影文字向上，当 shadowOffsetY 值为负数时产生的阴影文字向下；shadowBlur 表示阴影模糊效果，值越大模糊越厉害。

此实例的源文件名是 myHtmlA059.html。

003　在画布上创建向左或向右的 3D 拉影文字

此实例主要通过设置阴影属性值实现在画布上创建向左或向右的 3D 拉影的特效文字。当在浏览器中显示该页面时，单击“在画布上创建向左 3D 拉影的特效文字”按钮将在下面显示向左 3D 拉影的特效文字“炫酷实例集锦”，如图 003-1 所示；单击“在画布上创建向右 3D 拉影的特效文字”按钮将在下面显示向右 3D 拉影的特效文字“炫酷实例集锦”，如图 003-2 所示。有关此实例的主要代码如下。



图　003-1

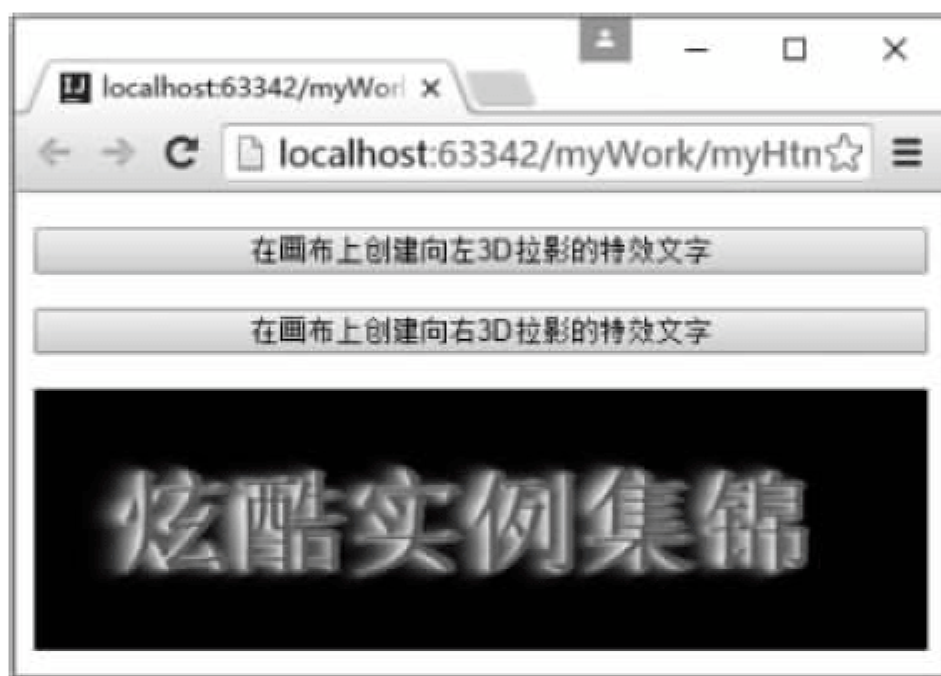


图　003-2

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    function myEffect(myDir) {
     var myCanvas = document.getElementById("myCanvas");
     var myContext = myCanvas.getContext('2d');
     myContext.font = '40pt 宋体';
```

```
        var myHeight = 0;
        myContext.fillStyle = "black";
        myContext.fillRect(0, myHeight, 410, 130);
        for (var i = 0; i < 10; i++) {
         myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
         if(myDir == "Left"){
          myContext.shadowOffsetX = i * 2;
         }else{
          myContext.shadowOffsetX = - i * 2;
         }
         myContext.shadowOffsetY = 0;
         myContext.shadowBlur = i * 2;
         myContext.fillStyle = "rgba(127, 127, 127, 1)";
         myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
        } }
        $("#myBtnLeft").click(function() {       //在画布上创建向左 3D 拉影的特效文字
         myEffect("Left");
        });
        $("#myBtnRight").click(function() {      //在画布上创建向右 3D 拉影的特效文字
         myEffect("Right");
      });});
    </script></head>
    <body><p><input type = "button" value = "在画布上创建向左 3D 拉影的特效文字" id = "myBtnLeft" style =
    "width:410px"></p>
    <p><input type = "button" value = "在画布上创建向右 3D 拉影的特效文字" id = "myBtnRight" style =
    "width:410px"></p>
    <canvas id = "myCanvas" width = "410" height = "120"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，shadowColor 表示设置画布的阴影颜色；shadowOffsetX 表示阴影相对于文字的水平距离，0 表示两者水平位置重合，当 shadowOffsetX 值为正数时产生的阴影文字向左，当 shadowOffsetX 值为负数时产生的阴影文字向右；shadowBlur 表示阴影模糊效果，值越大模糊越厉害。

此实例的源文件名是 myHtmlA060.html。

004　在画布上创建模糊阴影的文字

此实例主要通过设置阴影属性值实现在画布上创建具有模糊阴影的特效文字。当在浏览器中显示该页面时，单击“在画布上创建模糊阴影的特效文字”按钮将在下面显示具有模糊阴影的特效文字“炫酷实例集锦”，如图 004-1 所示。有关此实例的主要代码如下。

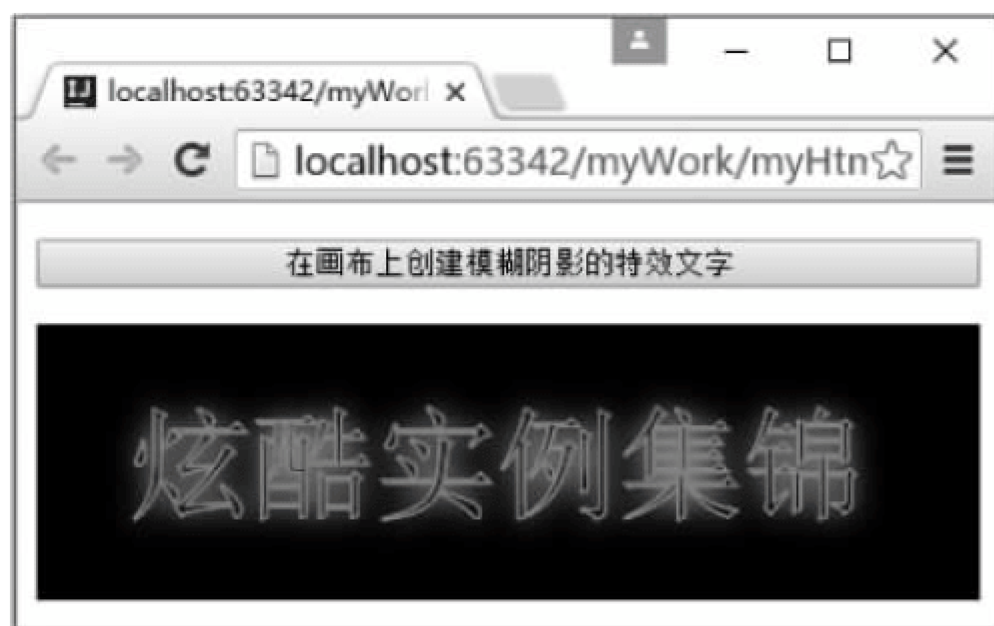


图　004-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {        //在画布上创建模糊阴影的特效文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    myContext.font = '40pt 宋体';
    var myHeight = 0;
    myContext.fillStyle = "black";
    myContext.fillRect(0, myHeight, 410, 130);
    myContext.shadowColor = "white";
    myContext.shadowOffsetX = 0;
    myContext.shadowOffsetY = 0;
    myContext.shadowBlur = 20;
    myContext.fillText("炫酷实例集锦", 40, 80);
    myContext.strokeStyle = "rgba(0, 255, 0, 1)";
    myContext.lineWidth = 1;
    myContext.strokeText("炫酷实例集锦", 40, 80);
  });});
</script></head>
<body><p><input type="button" value="在画布上创建模糊阴影的特效文字" id="myBtn" style="width:
410px"></p>
<canvas id="myCanvas" width="410" height="120"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，shadowColor 表示设置画布的阴影颜色；shadowOffsetX 表示阴影相对于文字的水平距离；shadowOffsetY 表示阴影相对于文字的垂直距离；shadowBlur 表示阴影模糊效果，值越大模糊越厉害。fillText()方法用于在画布上绘制填色的文字。strokeText()方法用于在画布上绘制文本(没有填色)，该方法的语法声明如下。

```
strokeText(text,x,y,maxWidth);
```

其中，参数 text 规定在画布上输出的文本；参数 x 表示开始绘制文本的 x 坐标位置(相对于画布)；参数 y 表示开始绘制文本的 y 坐标位置(相对于画布)；参数 maxWidth 是可选参数，表示允许的最大文本宽度，以像素计算。

此实例的源文件名是 myHtmlA061.html。

005　在画布上创建边缘羽化的文字

此实例主要通过设置 shadowColor、shadowOffsetX、shadowOffsetY、shadowBlur 等阴影属性值基于 3D 拉影效果在 4 个方向上重复，从而实现在画布上创建字体边缘羽化的特效文字。当在浏览器中显示该页面时，单击"在画布上创建边缘羽化的特效文字"按钮将在下面显示具有边缘羽化的特效文字"炫酷实例集锦"，如图 005-1 所示。有关此实例的主要代码如下。



图　005-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {      //在画布上创建边缘羽化的特效文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    myContext.font = '40pt 宋体';
    var myHeight = 0;
    myContext.fillStyle = "green";
    myContext.fillRect(0, myHeight, myCanvas.width, 130);
    for(var i = 0; i<10; i++){
     //设置画布的阴影颜色
     myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
     //设置阴影相对于文字的水平距离
     myContext.shadowOffsetX = 0;
     //设置阴影相对于文字的垂直距离
     myContext.shadowOffsetY = - i * 2;
     //设置阴影模糊效果,值越大模糊越厉害
     myContext.shadowBlur = i * 2;
     //设置文字颜色
     myContext.fillStyle = "rgba(127, 127, 127, 1)";
     myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
    }
    for(var i = 0; i<10; i++){
     myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
     myContext.shadowOffsetX = 0;
     myContext.shadowOffsetY = i * 2;
     myContext.shadowBlur = i * 2;
     myContext.fillStyle = "rgba(127, 127, 127, 1)";
     myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
    }
    for(var i = 0; i<10; i++){
     myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
     myContext.shadowOffsetX = i * 2;
     myContext.shadowOffsetY = 0;
     myContext.shadowBlur = i * 2;
     myContext.fillStyle = "rgba(127, 127, 127, 1)";
     myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
    }
    for(var i = 0; i<10; i++){
     myContext.shadowColor = "rgba(255, 255, 255," + ((10 - i)/10) + ")";
     myContext.shadowOffsetX = - i * 2;
     myContext.shadowOffsetY = 0;
     myContext.shadowBlur = i * 2;
     myContext.fillStyle = "rgba(127, 127, 127, 1)";
     myContext.fillText("炫酷实例集锦", 40, 80 + myHeight);
  } });});
</script></head>
<body><p><input type="button" value="在画布上创建边缘羽化的特效文字" id="myBtn" style="width:
410px"></p>
<canvas id="myCanvas" width="410" height="120"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。此实例的源文件名是 myHtmlA062.html。

006　在画布上创建空心线条的文字

此实例主要通过设置 strokeStyle 和 fillStyle 属性实现在画布上创建空心线条的特效文字。当在浏览器中显示该页面时，单击“在画布上创建空心线条的特效文字”按钮将在下面显示具有空心线条的特效文字“炫酷实例集锦”，如图 006-1 所示。有关此实例的主要代码如下。



图　006-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {                      //在画布上创建空心线条的特效文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    myContext.font = '40pt 宋体';
    var myHeight = 0;
    myContext.fillStyle = "white";                      //设置背景颜色
    myContext.fillRect(0, myHeight, myCanvas.width, 130);//绘制背景矩形
    myContext.lineWidth = 3;
    myContext.strokeStyle = 'green';                    //设置描边颜色
    myContext.strokeText('炫酷实例集锦', 40, 80);        //绘制描边文本
    myContext.fillStyle = 'white';                      //设置填充颜色
    myContext.fillText('炫酷实例集锦', 40, 80);          //绘制填充文本
  });});
</script></head>
<body><p><input type = "button" value = "在画布上创建空心线条的特效文字" id = "myBtn" style = "width:
410px"></p>
<canvas id = "myCanvas" width = "410" height = "120"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，fillStyle 属性用于设置或返回填充绘画的颜色、渐变或模式；strokeStyle 属性用于设置或返回笔触的颜色、渐变或模式；strokeText()方法用于在画布上绘制文本(没有填色)；fillText()方法用于在画布上绘制填色的文本。

此实例的源文件名是 myHtmlA063.html。

007　在画布上绘制半透明阴影文字

此实例主要通过设置 shadowColor 等阴影属性值实现在画布上绘制半透明的立体阴影文字。当在浏览器中显示该页面时，单击“绘制半透明的阴影字体”按钮将在下面显示半透明立体阴影效果的

文字"炫酷实例集锦",如图 007-1 所示。有关此实例的主要代码如下。



图 007-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function(){
   $("#myBtn").click(function() {       //绘制半透明的阴影字体
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext("2d");
    myContext.shadowOffsetX = 2;
    myContext.shadowOffsetY = 2;
    myContext.shadowBlur = 2;
    //0.5 表示半透明,可能值在 0 到 1 之间
    myContext.shadowColor = "rgba(0, 0, 0, 0.5)";
    myContext.fillStyle = "#000";
    myContext.font = "60px 宋体";
    myContext.fillText("炫酷实例集锦", 20, 50);
  });});
 </script></head>
<body><p><input type="button" value="绘制半透明的阴影字体" id="myBtn" style="width:400px"></p
>
<canvas id="myCanvas" width="410" height="410"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,shadowColor 表示设置画布的阴影颜色;shadowOffsetX 表示阴影相对于文字的水平距离;shadowOffsetY 表示阴影相对于文字的垂直距离;shadowBlur 表示阴影模糊效果,值越大模糊越厉害;fillStyle 属性用于设置或获取填充绘画的颜色、渐变或模式;font 属性用于设置或获取画布上文本内容的当前字体属性;fillText()方法用于在画布上绘制填色的文字,文字的默认颜色是黑色。

此实例的源文件名是 myHtmlA069.html。

008 在画布上绘制左右渐变的文字

此实例主要通过使用 createLinearGradient()方法创建线性渐变对象,从而实现在画布上绘制从左至右的线性渐变文字。当在浏览器中显示该页面时,单击"绘制从左至右的线性渐变文字"按钮将在下面显示从左至右的线性渐变文字"炫酷实例集锦",如图 008-1 所示。有关此实例的主要代码如下。



图　008-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {       //绘制从左至右的线性渐变文字
    var myCanvas=document.getElementById("myCanvas");
    var myContext=myCanvas.getContext("2d");
    myContext.shadowOffsetX=2;
    myContext.shadowOffsetY=2;
    myContext.shadowBlur=2;
    //0.5 表示半透明,可能值在 0 到 1 之间
    myContext.shadowColor="rgba(0, 0, 0, 0.5)";
    var myGradient=myContext.createLinearGradient(0,0,400,0);
    //设置线性渐变的开始颜色为红色
    myGradient.addColorStop(0,"red");
    //设置线性渐变的结束颜色为绿色
    myGradient.addColorStop(1,"green");
    myContext.fillStyle=myGradient;
    myContext.font="60px 宋体";
    myContext.fillText("炫酷实例集锦", 20, 50);
  });});
 </script></head>
<body><p><input type="button" value="绘制从左至右的线性渐变文字" id="myBtn" style="width:
400px"></p>
<canvas id="myCanvas" width="410" height="410"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，createLinearGradient()方法用于创建线性的渐变对象，渐变对象可用于填充矩形、圆形、线条、文本等，该方法的语法声明如下。

```
createLinearGradient(x0,y0,x1,y1)
```

其中，参数 x0 表示渐变开始点的 x 坐标；参数 y0 表示渐变开始点的 y 坐标；参数 x1 表示渐变结束点的 x 坐标；参数 y1 表示渐变结束点的 y 坐标。

当使用 createLinearGradient()方法成功创建一个线性渐变对象后，通常需要用 addColorStop()方法设置渐变过程需要使用的颜色，addColorStop()方法的语法声明如下。

```
addColorStop(stop,color)
```

其中，参数 stop 是 0.0～1.0 的值，表示渐变中开始与结束之间的位置；参数 color 表示在结束位置显示的 CSS 颜色值。

使用 createLinearGradient()方法创建的线性渐变对象可以直接赋值给画布的 fillStyle 属性，从而使画布的呈现内容出现渐变效果。fillStyle 属性设置或获取用于填充绘画的颜色、渐变或模式，它

支持 3 种属性值，即 color(指示绘图填充色的 CSS 颜色值，默认值是＃000000)、gradient(用于填充绘图的线性或放射性渐变对象)、pattern(用于填充绘图的 pattern 对象)。

此实例的源文件名是 myHtmlA070.html。

009 在画布上绘制扁平或细长的文字

此实例主要通过使用 scale()方法对画布进行水平拉伸或垂直拉伸，从而实现绘制扁平文字或细长文字。当在浏览器中显示该页面时，单击"绘制扁平文字"按钮将在下面显示经过水平拉伸放大的扁平文字，如图 009-1 所示；单击"绘制细长文字"按钮将在下面显示经过垂直拉伸放大的细长文字，如图 009-2 所示。有关此实例的主要代码如下。



图 009-1



图 009-2

```
<!doctype html><html><head><meta charset = UTF - 8 >
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtnHorizontal").click(function() {        //绘制扁平文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext("2d");
    myContext.clearRect(0,0,400,200);
    myContext.save();
    myContext.font = "30px 宋体";
    //按照 2 倍在水平方向上拉伸
    myContext.scale(2,1);
    myContext.fillText("炫酷实例集锦", 10, 30);
    myContext.restore();
   });
   $("#myBtnVertical").click(function() {          //绘制细长文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext("2d");
    myContext.clearRect(0,0,400,200);
    myContext.save();
    myContext.font = "30px 宋体";
    //按照 4 倍在垂直方向上拉伸
    myContext.scale(1,4);
    myContext.fillText("HTML5 炫酷实例集锦", 30, 30);
    myContext.restore();
  });});
```

```
</script></head>
<body><p><input type="button" value="绘制扁平文字" id="myBtnHorizontal" style="width: 190px">
<input type="button" value="绘制细长文字" id="myBtnVertical" style="width: 190px"></p>
<canvas id="myCanvas" width="410" height="310"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，save()方法用于保存当前环境的状态；restore()方法用于恢复之前保存过的路径状态和属性；scale()方法用于缩放当前绘图，使其更大或更小，如果对绘图进行缩放，之后所有的绘图也会被缩放，定位也会被缩放，例如 scale(2,2)，表示绘图将定位于距离画布左上角 2 倍远的位置，scale(2,1)表示水平方向拉伸 2 倍、垂直方向不变，scale(1,4)表示水平方向不变、垂直方向拉伸 4 倍；fillText()方法用于在画布上绘制填色的文本。

此实例的源文件名是 myHtmlA072.html。

010　在画布上使用图像填充文字线条

此实例主要通过使用 createPattern()方法设置填充的图像，从而实现在画布上使用图像填充显示文字的线条。当在浏览器中显示该页面时，单击"使用图像填充显示文字的线条"按钮将在下面显示用图像填充线条的文字"炫酷实例集锦"，如图 010-1 所示。有关此实例的主要代码如下。



图　010-1

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {        //使用图像填充显示文字的线条
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext("2d");
    var myImage = new Image();
    myImage.src = "img/a048.jpg";
    myImage.onload = function() {
     myContext.shadowOffsetX = 2;
     myContext.shadowOffsetY = 2;
     myContext.shadowBlur = 2;
     myContext.shadowColor = "rgba(0, 0, 0, 0.5)";
     //设置图像平铺模式
     var myPattern = myContext.createPattern(myImage, "repeat");
     //设置图像显示范围
     myContext.rect(0, 0, 400, 200);
     //设置填充样式
     myContext.fillStyle = myPattern;
```

```
        myContext.font = "60px 宋体";
        myContext.fillText("炫酷实例集锦", 20, 50);
   };});});
  </script></head>
<body><p><input type = "button" value = "使用图像填充显示文字的线条" id = "myBtn" style = "width:
400px"></p>
<canvas id = "myCanvas" width = "410" height = "410"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，createPattern()方法用于在指定的方向内重复指定的元素，这些元素可以是图片、视频或者其他 canvas 元素，被重复的元素可用于绘制/填充矩形、圆形或线条等。由于使用 createPattern()方法创建的模式对象可以直接赋值给画布的 fillStyle 属性，因而可以使画布的呈现内容出现填充应有的效果。

此实例的源文件名是 myHtmlA071.html。

011　在画布上移动鼠标指针实现文字涂鸦

此实例主要为画布添加 mousedown、mousemove、mouseup 等事件响应方法，并使用 HTML5 的直线绘制方法，从而实现在自定义画布上通过移动鼠标指针进行文字涂鸦。当在 Google Chrome 浏览器中显示该页面时，在灰色的自定义画布上即可移动鼠标指针进行文字涂鸦，如图 011-1 所示。有关此实例的主要代码如下。

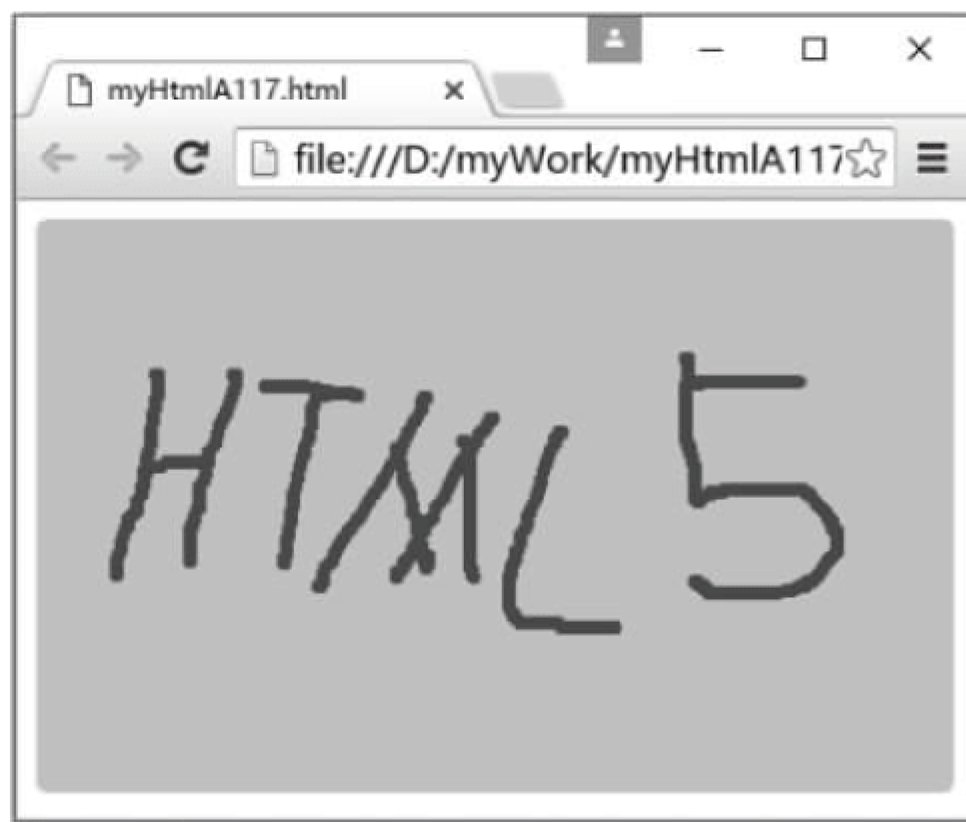


图　011-1

```
<!doctype html><html><head><meta charset = "UTF-8">
<script src = "js/jquery-3.1.1.min.js"></script><script type = "text/javascript">
   $(function() {
    myContext = document.getElementById('myCanvas').getContext("2d");
    var myClickX = new Array();
    var myClickY = new Array();
    var myClickDrag = new Array();
    var myPaint;
    function addClick(x, y, dragging) {            //记录鼠标指针位置
     myClickX.push(x);
     myClickY.push(y);
     myClickDrag.push(dragging);
```

```
    }
    function redraw() {                              //将记录的鼠标指针位置连线
     myContext.strokeStyle = "green";
     myContext.lineJoin = "round";
     myContext.lineWidth = 5;
     for (var i = 0; i < myClickX.length; i++) {
      myContext.beginPath();
      if (myClickDrag[i] && i) {
        myContext.moveTo(myClickX[i - 1], myClickY[i - 1]);
      } else {
        myContext.moveTo(myClickX[i] - 1, myClickY[i]);
      }
     myContext.lineTo(myClickX[i], myClickY[i]);
     myContext.closePath();
     myContext.stroke();
   } }
   $ ('#myCanvas').mousedown(function(e) {             //重新开始绘制
     myPaint = true;
     addClick(e.pageX - this.offsetLeft, e.pageY - this.offsetTop);
     redraw();
   });
   $ ('#myCanvas').mousemove(function(e) {             //继续绘制
    if (myPaint) {
      addClick(e.pageX - this.offsetLeft, e.pageY - this.offsetTop, true);
      redraw();
    } });
   $ ('#myCanvas').mouseup(function(e) {               //绘制结束
    myPaint = false;
   });});
  </script>
  <style type = "text/css">
   #myCanvas { background-color: lightgrey; border-radius: 5px; }
  </style></head>
 <body><canvas id = "myCanvas" width = "400" height = "250"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$('#myCanvas').mousedown(function(e){})方法用于响应鼠标按下事件，$('#myCanvas').mousemove(function(e) { })方法用于响应鼠标移动事件，$('#myCanvas').mouseup(function(e){})方法用于响应鼠标抬起(按键释放)事件；myContext.beginPath()用于开始一条路径或重置当前的路径；myContext.moveTo(myClickX[i]－1，myClickY[i])用于把路径移动到画布中的指定点，不创建线条；myContext.lineTo(myClickX[i]，myClickY[i])用于添加一个新点，然后创建从该点到画布中最后指定点的线条(该方法并不会创建线条)；myContext.closePath()创建从当前点到开始点的路径；myContext.stroke()实际地绘制出通过moveTo()和lineTo()方法定义的路径，默认的填充颜色是黑色，此实例自定义填充颜色为绿色。

此实例的源文件名是myHtmlA117.html。

012　将画布上的文字、图像等保存到本地

此实例主要通过使用toDataURL()方法实现把画布上显示的图像、文字等所有内容以文件的形式保存到本地计算机中。当在浏览器中显示该页面时，单击“在画布上显示图像和文字”按钮将在画

布上显示一幅图像和一行文字；单击“将图像和文字保存到本地”按钮，则浏览器执行下载动作，如图 012-1 所示，图像文件将被保存到本地计算机的下载文件夹中。有关此实例的主要代码如下。



图 012-1

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtnShow").click(function() {        //在画布上显示图像和文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    var myImage = new Image();
    myImage.src = 'img/a040.jpg';
    myImage.onload = function() {            //绘制图像
     myContext.drawImage(myImage,0, 0,myCanvas.width,myCanvas.height);
     myContext.font = '16pt 宋体';
     myContext.lineWidth = 3;
     myContext.strokeStyle = 'green';
     myContext.strokeText('希拉里.克林顿', 140, 50);
     myContext.fillStyle = 'white';
     myContext.fillText('希拉里.克林顿', 140, 50);
    } });
   $("#myBtnSave").click(function(){         //将图像和文字保存到本地
    var myCanvas = document.getElementById("myCanvas");
    var myImage = myCanvas.toDataURL("image/png").
                          replace("image/png", "image/octet - stream");
    window.location.href = myImage; });});
 </script></head>
<body><p><input type = "button" value = "在画布上显示图像和文字" id = "myBtnShow" style =
"width:410px"></p>
<p><input type = "button" value = "将图像和文字保存到本地" id = "myBtnSave" style = "width:410px"></p>
<canvas id = "myCanvas" style = "width:410px;height:200px" ></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，toDataURL()方法用于把画布上显示的图像、文字等所有内容保存为图像。

此实例的源文件名是 myHtmlA065.html。

013　使用 SVG 实现文字在圆弧上显示

此实例主要使用 SVG 设置不同样式的路径并在路径上创建渐变色文字，从而实现在直线上和在半圆弧上显示渐变色文字的效果。当在 Google Chrome 浏览器中显示该页面时，单击“以弯曲路径显示文字”按钮，文字在圆弧上的显示效果如图 013-1 所示；单击“以水平路径显示文字”按钮，文字在直线上的显示效果如图 013-2 所示。有关此实例的主要代码如下。



图　013-1



图　013-2

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#myBtnCurve").click(function(){          //以弯曲路径显示文字
    $("svg.texts text.text-1").css("display","initial");
    $("svg.texts text.text-2").css("display","none");
   });
   $("#myBtnHorizontal").click(function(){    //以水平路径显示文字
```

```
      $("svg.texts text.text-1").css("display","none");
      $("svg.texts text.text-2").css("display","initial");
    });
    $("#myBtnCurve").click();
  });
  </script>
  <style type="text/css">
   .svg-defs { position: absolute;}
   .texts { text-align: center; width: 500px; height: 300px;}
   .text-1 { fill: url(#p-stripes);font-size: 48px;font-weight: bold;}
   .text-2 { fill: url(#p-stripes);font-size: 38px;font-weight: bold;}
   button{ width:200px;}
  </style></head>
<body><div align="center">
  <p><button id="myBtnCurve">以弯曲路径显示文字</button>
     <button id="myBtnHorizontal">以水平路径显示文字</button></p>
  <svg class="svg-defs"><g><defs>
     <path id="MyPath1" d="M 110 300 A100,110 0 0,1 505,300"
      transform="scale(0.8)"/>
     <path id="MyPath2" d="M 50 100 A100,0 0 0,1 600,100"
      transform="scale(0.8)"/>
     <linearGradient id="MyGradient" x1="0" y1="0" x2="100%" y2="0%">
      <stop offset="0%" stop-color="crimson" />
      <stop offset="10%" stop-color="red" />
      <stop offset="20%" stop-color="orangered" />
      <stop offset="30%" stop-color="orange" />
      <stop offset="40%" stop-color="gold" />
      <stop offset="50%" stop-color="yellowgreen" />
      <stop offset="60%" stop-color="green" />
      <stop offset="70%" stop-color="steelblue" />
      <stop offset="80%" stop-color="mediumpurple" />
      <stop offset="90%" stop-color="purple" /></linearGradient>
     <pattern id="p-stripes" patternUnits="userSpaceOnUse" width="200" height="200" viewbox="0 0
200 200">
       <rect width="200" height="200" fill="url(#MyGradient)"/></pattern></defs></g></svg>
  <svg class="texts">
   <text class="text-1" transform="translate(0)">
    <textPath xlink:href="#MyPath1">HTML5 + CSS3 炫酷实例集锦</textPath></text>
   <text class="text-2" transform="translate(0)">
    <textPath xlink:href="#MyPath2">HTML5 + CSS3 炫酷实例集锦</textPath></text></svg></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，标签<svg></svg>中的代码是符合SVG规范的代码。SVG是可缩放矢量图形(Scalable Vector Graphics)，基于可扩展标记语言，是用于描述二维矢量图形的一种图形格式，它由万维网联盟制定，是一个开放标准。<path id="MyPath1" d="M 110 300 A100,110 0 0,1 505,300" transform="scale(0.8)"/>用于创建半圆弧的路径，d属性值用于定义路径上的关键点坐标，scale(0.8)表示将原始路径缩小至80%。<linearGradient></linearGradient>用于设置各种颜色在路径上的分布位置，offset值以百分比表示。

此实例的源文件名是myHtmlA182.html。

014 使用 SVG 实现文字绕三角形显示

此实例主要使用 SVG 设置路径为三角形并在路径上创建渐变色文字，从而实现文字围绕三角形周边显示的效果。当在 Google Chrome 浏览器中显示该页面时，文字围绕三角形周边显示的效果如图 014-1 所示。有关此实例的主要代码如下。



图 014-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .svg-defs { position: absolute;}
  .texts { text-align: center; width: 430px; height: 300px;}
  /*设置文字的基本样式*/
  .myText { fill: url(#p-stripes);font-size:29px;font-weight: bold;}
 </style></head>
<body><div align="center">
 <svg class="svg-defs"><g><defs>
    <!--设置路径是一个三角形-->
    <path id="MyPath1" d="M 250 20 L 70 370 L 450 370 Z" transform="scale(0.8)"/>
    <!--设置文字在三角形路径上的渐变颜色-->
    <linearGradient id="MyGradient" x1="0" y1="0" x2="100%" y2="0%">
     <stop offset="0%" stop-color="crimson" />
     <stop offset="10%" stop-color="red" />
     <stop offset="20%" stop-color="orangered" />
     <stop offset="30%" stop-color="orange" />
     <stop offset="40%" stop-color="gold" />
     <stop offset="50%" stop-color="yellowgreen" />
     <stop offset="60%" stop-color="green" />
     <stop offset="70%" stop-color="steelblue" />
     <stop offset="80%" stop-color="mediumpurple" />
     <stop offset="90%" stop-color="purple" /></linearGradient>
    <pattern id="p-stripes" patternUnits="userSpaceOnUse" width="200" height="200" viewbox="0 0
200 200">
```

```
        <rect width = "200" height = "200" fill = "url(#MyGradient)"/>
      </pattern></defs></g></svg>
    <svg class = "texts">
     <text class = "myText" transform = "translate(0)">
      <textPath xlink:href = "#MyPath1">关关雎鸠,在河之洲。窈窕淑女,君子好逑。参差荇菜,左右流之。
</textPath></text>
    </svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,< path id="MyPath1" d="M 250 20 L 70 370 L 450 370 Z" transform="scale(0.8)"/>用于设置 SVG 的路径是三角形,其中 250 20 表示三角形的上顶点,70 370 表示三角形的左下顶点,450 370 表示三角形的右下顶点。M、L、Z 实际上是方法(命令)的缩写。下面的命令可用于路径数据:

```
M = moveto
L = lineto
H = horizontal lineto
V = vertical lineto
C = curveto
S = smooth curveto
Q = quadratic Bezier curve
T = smooth quadratic Bezier curveto
A = elliptical Arc
Z = closepath
```

以上所有命令均允许写成小写字母,大写表示绝对定位,小写表示相对定位。

此实例的源文件名是 myHtmlA187.html。

015 使用 SVG 实现文字沿曲线显示

此实例主要使用 SVG 设置路径为三次贝塞尔曲线,并在路径上创建渐变色文字,从而实现文字沿着三次贝塞尔曲线显示的效果。当在 Google Chrome 浏览器中显示该页面时,文字沿着三次贝塞尔曲线显示的效果如图 015-1 所示。有关此实例的主要代码如下。



图 015-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .svg-defs { position: absolute;}
  .texts { text-align: center; width: 460px; height:250px;}
  /*设置文字的基本样式*/
  .myText { fill: url(#p-stripes);font-size:18px;font-weight: bold;}
 </style></head>
<body><div align="center"><svg class="svg-defs"><g><defs>
    <!--设置路径是三次贝塞尔曲线-->
    <path id="MyPath1" d="M100,200 C100,100 250,100 250,200 S400,300 400,200 " transform=
"scale(1.2)"/>
    <!--设置文字在三次贝塞尔曲线路径上的渐变颜色-->
    <linearGradient id="MyGradient" x1="0" y1="0" x2="100%" y2="0%">
     <stop offset="0%" stop-color="crimson" />
     <stop offset="10%" stop-color="red" />
     <stop offset="20%" stop-color="orangered" />
     <stop offset="30%" stop-color="orange" />
     <stop offset="40%" stop-color="gold" />
     <stop offset="50%" stop-color="yellowgreen" />
     <stop offset="60%" stop-color="green" />
     <stop offset="70%" stop-color="steelblue" />
     <stop offset="80%" stop-color="mediumpurple" />
     <stop offset="90%" stop-color="purple" /></linearGradient>
    <pattern id="p-stripes" patternUnits="userSpaceOnUse" width="200" height="200" viewbox="0 0
200 200">
     <rect width="200" height="200" fill="url(#MyGradient)"/></pattern></defs></g></svg>
 <svg class="texts">
  <text class="myText" transform="translate(-60,-100)">
   <textPath xlink:href="#MyPath1">关关雎鸠,在河之洲。窈窕淑女,君子好逑。参差荇菜,左右流之。
</textPath></text></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<path id="MyPath1" d="M100,200 C100,100 250,100 250,200 S400,300 400,200 " transform="scale(1.2)"/>中的 d 属性值用于设置三次贝塞尔曲线的控制点;<text class="myText" transform="translate(-60,-100)">中的 transform 属性值表示根据 translate(-60,-100)的参数值平移<text>标签。

此实例的源文件名是 myHtmlA188.html。

016　使用 SVG 实现文字沿跑道线显示

此实例主要使用 SVG 设置路径为体育场跑道线,并在路径上创建渐变色文字,从而实现文字沿着体育场跑道线显示的效果。当在 Google Chrome 浏览器中显示该页面时,文字沿着体育场跑道线显示的效果如图 016-1 所示。有关此实例的主要代码如下。



图　016-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  .myText { fill: url(#p-stripes); font-size: 20px; font-weight: bold;}
 </style></head>
<body><div align = "center">
 <svg width = "100%" height = "100%"><defs>
   <!-- 设置路径的形状(体育场跑道线)-->
   <path id = "MyPath1" d = "M50,90 C0,90 0,30 50,30 L150,30 C200,30 200,90 150,90 z" transform = "scale
(2)"/>
   <!-- 设置文字在路径上的渐变颜色 -->
   <linearGradient id = "MyGradient" x1 = "0" y1 = "0" x2 = "100%" y2 = "0%">
    <stop offset = "0%" stop-color = "crimson"/>
    <stop offset = "10%" stop-color = "red"/>
    <stop offset = "20%" stop-color = "orangered"/>
    <stop offset = "30%" stop-color = "orange"/>
    <stop offset = "40%" stop-color = "gold"/>
    <stop offset = "50%" stop-color = "yellowgreen"/>
    <stop offset = "60%" stop-color = "green"/>
    <stop offset = "70%" stop-color = "steelblue"/>
    <stop offset = "80%" stop-color = "mediumpurple"/>
    <stop offset = "90%" stop-color = "purple"/> </linearGradient>
   <pattern id = "p-stripes" patternUnits = "userSpaceOnUse" width = "200" height = "200" viewbox = "0 0
200 200">
     <rect width = "200" height = "200" fill = "url(#MyGradient)"/></pattern></defs>
  <g><text class = "myText" transform = "translate(10, -30),scale(0.9)">
   <textPath xlink:href = "#MyPath1">关关雎鸠,在河之洲。窈窕淑女,君子好逑。参差荇菜,左右流之。窈
窕淑女,寤寐求之。</textPath></text></g></svg></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<path id="MyPath1" d="M50,90 C0,90 0,30 50,30 L150,30 C200,30 200,90 150,90 z" transform="scale(2)"/>中的 d 属性值用于设置体育场跑道线的控制点,transform="scale(2)"表示将体育场跑道线放大两倍;<text class="myText" transform="translate(10,-30),scale(0.9)">中的 transform 属性值表示根据 translate(10,-30),scale(0.9)的参数值平移和缩小<text>标签;fill: url(#p-stripes)表示文字的颜色使用SVG 创建的渐变色填充。

此实例的源文件名是 myHtmlA216.html。

017 在 SVG 中对线条进行加粗或瘦身

此实例主要在 SVG 中设置 feMorphology 滤镜的 operator 属性值,从而实现对文本的线条进行增肥和瘦身处理。当在 Google Chrome 浏览器中显示该页面时将显示原始的文本,单击"使用瘦身的线条显示文本"按钮,则文本线条经过瘦身处理之后的效果如图 017-1 所示;单击"使用增肥的线条显示文本"按钮,则文本线条经过增肥处理之后的效果如图 017-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF-8">
<script src = "js/jquery-3.1.1.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myBtn1").click(function() {      //使用瘦身的线条显示文本
    $("g").attr("filter", "url(#myErode)");
```

```
    });
    $("#myBtn2").click(function() {                //使用增肥的线条显示文本
     $("g").attr("filter", "url(#myDilate)"); });});
  </script>
  <style type="text/css">
   svg {margin-top: 20px; position: absolute;left: calc(50% - 200px);}
   div { margin-top: 2px;margin-left: 2px;}
   button {width: 195px;}
   .myText { font-size: 64px; font-weight: bold; }
  </style></head>
<body><div align="center">
  <button id="myBtn1">使用瘦身的线条显示文本</button>
  <button id="myBtn2">使用增肥的线条显示文本</button></div>
<div><svg width="400" height="275"><defs>
    <filter id="myErode"><feMorphology operator="erode" in="SourceGraphic" radius=
"1.95" /></filter>
    <filter id="myDilate"><feMorphology operator="dilate" in="SourceGraphic" radius=
"2.9" /></filter></defs>
   <g><text x="0" y="60" class="myText">炫酷实例集锦</text></g></svg></div> </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feMorphology 滤镜用于对源图形执行 fattening(增肥)或者 thinning(瘦身)，当设置 feMorphology 滤镜的 operator 属性值为 erode(侵蚀)时，该滤镜产生的是瘦身特效；当设置 feMorphology 滤镜的 operator 属性值为 dilate(膨胀)时，该滤镜产生的是增肥特效，瘦身和增肥特效的数值由 feMorphology 滤镜的 radius 值确定。

此实例的源文件名是 myHtmlA234.html。



图 017-1



图 017-2

018 使用 SVG 描边创建空心线条文字

此实例主要通过使用 SVG 的与描边相关的属性创建空心线条文字。当在 Google Chrome 浏览器中显示该页面时将显示默认的实心文字，单击“显示空心文字”按钮则显示设置描边属性之后创建的空心线条文字，如图 018-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtn1").click(function() {     //显示实心文字
     $("text").attr("fill", "");
     $("text").attr("stroke", "");
```

```
    });
    $("#myBtn2").click(function() {                //显示空心文字
      $("text").attr("fill", "white");
      $("text").attr("stroke", "green");
      $("text").attr("stroke-dasharray", "2"); });});
  </script>
  <style type="text/css">
   .mySVG { width: 410px; height: 410px;}
   text{font-family: simsun,Tahoma,Helvetica,Arial,SimHei,sans-serif;
        font-size: 100px; font-weight: bold;}
   button { width: 195px;}
  </style></head>
 <body><div align="center">
  <button id="myBtn1">显示实心文字</button>
  <button id="myBtn2">显示空心文字</button></div>
 <div align="center"><svg class="mySVG">
   <text x="1" y="120" >实例集锦</text></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，stroke-dasharray、fill、stroke 几个属性用于设置描边样式。在 SVG 中，与描边相关的属性如下：

（1）stroke 属性。该属性用于定义一条线、文本或元素的轮廓颜色。

（2）stroke-width 属性。该属性用于定义一条线、文本或元素的轮廓厚度。

（3）stroke-linecap 属性。该属性用于描边端点表现形式，例如圆角、方形等。

（4）stroke-dasharray 属性。该属性用于创建虚线。

在 stroke-dasharray 属性中使用一个参数绘制虚线时表示一段虚线长度和每段虚线之间的间距，例如 stroke-dasharray='10'；在 stroke-dasharray 属性中使用两个参数绘制虚线时，一个参数表示长度，一个参数表示间距，例如 stroke-dasharray='10, 10'；在使用 4 个参数绘制虚线时则是两个参数情形的叠加，例如 stroke-dasharray='10, 10, 5, 5'。

另外还有其他的描边属性，例如 stroke-dashoffset、stroke-linejoin、stroke-opacity 等。

此实例的源文件名是 myHtmlA220.html。



图 018-1

019 使用 SVG 描边创建渐变空心文字

此实例主要在 SVG 中设置 stroke 属性为渐变色，从而创建渐变色的空心文字。当在 Google Chrome 浏览器中显示该页面时将显示默认的实心文字，单击“显示渐变色空心文字”按钮则创建渐变

色空心文字，效果如图 019-1 所示。有关此实例的主要代码如下。



图　019-1

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {        //显示实心文字
    $("text").attr("fill", "");
    $("text").attr("stroke", "");
   });
   $("#myBtn2").click(function() {        //显示渐变色空心文字
    $("text").attr("fill", "white");
    $("text").attr("stroke", "url(#myGradient)"); });});
 </script>
 <style type="text/css">
  .mySVG { width: 410px; height: 410px; }
  text{font-family: simsun,Tahoma,Helvetica,Arial,SimHei,sans-serif;
       font-size: 64px;font-weight: bold;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示实心文字</button>
 <button id="myBtn2">显示渐变色空心文字</button></div>
<div align="center"><svg class="mySVG"><defs>
   <linearGradient id="myGradient" x1="0%" y1="0%" x2="100%" y2="0%">
    <stop offset="0%" stop-color="red" />
    <stop offset="100%" stop-color="green" /></linearGradient></defs>
  <text x="10" y="100">炫酷实例集锦</text></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，stroke 属性用于定义文本或元素的轮廓颜色，即通常所说的描边，如果设置该属性值为渐变色，则文本或元素的轮廓也呈现渐变色。

此实例的源文件名是 myHtmlA221.html。

020　使用 SVG 滤镜创建扩散阴影文字

此实例主要通过使用 SVG 的 feGaussianBlur、feOffset、feFlood、feComposite 等滤镜创建具有模糊扩散阴影效果的文字。当在 Google Chrome 浏览器中显示该页面时，使用这些滤镜创建的特效文字效果如图 020-1 所示。有关此实例的主要代码如下。



图 020-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px); }
 </style></head>
<body><svg width="100%" height="100%"><defs>
  <filter id="myFilter" filterUnits="objectBoundingBox" x="-10%" y="-10%" width="150%"
height="150%">
   <!--产生模糊扩散特效-->
   <feGaussianBlur in="SourceAlpha" stdDeviation="3" result="blurredAlpha"/>
   <!--产生阴影偏移特效-->
   <feOffset in="blurredAlpha" dx="4" dy="4" result="offsetBlurredAlpha"/>
   <feFlood result="flooded" style="flood-color:darkgreen;flood-opacity:0.85"/>
   <!--<feComposite in="flooded" operator="atop" in2="offsetBlurredAlpha" result="coloredShadow"/>-->
   <!--设置 in 和 in2 的组合模式-->
   <feComposite in="flooded" operator="in" in2="offsetBlurredAlpha" result="coloredShadow"/>
   <feComposite in="SourceGraphic" in2="coloredShadow" operator="over"/>
  </filter></defs>
  <text x="0px" y="65px" style="fill:rgb(0,0,0); font-size:50px;" filter="url(#myFilter)">炫酷
应用实例集锦</text></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<filter></filter>标签中的feGaussianBlur、feOffset、feFlood、feComposite、feComposite 滤镜必须按照此顺序创建,因为feGaussianBlur 滤镜创建的 result="blurredAlpha"将被作为 feOffset 滤镜的输入项 in="blurredAlpha",以此类推。<feGaussianBlur in="SourceAlpha" stdDeviation="3" result="blurredAlpha"/>中的 stdDeviation="3"表示模糊半径,此值越大模糊范围越大。<feOffset in="blurredAlpha" dx="4" dy="4" result="offsetBlurredAlpha"/>中的 dx、dy 表示阴影偏移值,可以为负数,例如<feOffset in="blurredAlpha" dx="-4" dy="-4" result="offsetBlurredAlpha"/>。

此实例的源文件名是 myHtmlA194.html。

021 使用 SVG 滤镜创建木雕和蚀刻文字

此实例主要通过使用 SVG 的 feDiffuseLighting、feDistantLight、feOffset 等滤镜创建具有木雕和蚀刻效果的特效文字。当在 Google Chrome 浏览器中显示该页面时,单击“使用 feDiffuseLighting 滤镜创建特效文字一”按钮则创建木雕文字,效果如图 021-1 所示;单击“使用 feDiffuseLighting 滤镜创建特效文字二”按钮则创建蚀刻文字,效果如图 021-2 所示。有关此实例的主要代码如下。



图　021-1



图　021-2

```
<!doctype html><html><head><meta charset = "UTF-8">
<script src = "js/jquery-3.1.1.min.js"></script><script type = "text/javascript">
  $(function() {
   //使用 feDiffuseLighting 滤镜创建特效文字一
    $("#myBtn1").click(function() {
     $("svg defs filter feDiffuseLighting").attr("diffuseConstant", "0.5");
   });
   //使用 feDiffuseLighting 滤镜创建特效文字二
    $("#myBtn2").click(function() {
     $("svg defs filter feDiffuseLighting").attr("diffuseConstant", "5");
  });});
 </script>
 <style type = "text/css">
  svg { margin-top: 10px;position: absolute;
        left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align = "center">
 <button id = "myBtn1">使用 feDiffuseLighting 滤镜创建特效文字一</button>
 <button id = "myBtn2">使用 feDiffuseLighting 滤镜创建特效文字二</button></div>
  <svg width = "100%" height = "100%"><defs><filter id = "myFilter" x = "0" y = "0">
   <feOffset dx = "2" dy = "5" width = "400px"/>
   <feDiffuseLighting surfaceScale = "5" diffuseConstant = "0.5" style = "lighting-color:white;">
    <feDistantLight azimuth = "135" elevation = "60"/>
   </feDiffuseLighting></filter></defs>
  <text x = "0px" y = "65px" style = "font-size:66px;" filter = "url(#myFilter)">炫酷实例集锦</text>
</svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feDiffuseLighting 滤镜主要通过 alpha 通道实现轮廓的凹凸效果，此实例通过调整其 diffuseConstant 属性值即得两种风格各异的文字特效；feDistantLight 滤镜主要用于控制光源角度，即通过 azimuth 和 elevation 属性值调整光源在 XYZ 坐标系中的角度。

此实例的源文件名是 myHtmlA196.html。

022　使用 SVG 的动画变换实现文字旋转

此实例主要在 SVG 的<text>标签中嵌入<animateTransform>标签，从而实现<text>标签代表的文字根据<animateTransform>标签指定的属性值进行旋转的效果。当在 Google Chrome 浏览器

中显示该页面时，"HTML5+CSS3 炫酷实例集锦"自动围绕中心旋转，如图 022-1 所示。有关此实例的主要代码如下。



图 022-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  .mySVG { width: 400px; height: 400px;}
  text {font-family: simsun, Tahoma, Helvetica, Arial, SimHei, sans-serif;
        font-size: 24px;font-weight: bold;}
 </style></head>
<body><div align = "center"><svg class = "mySVG">
  <text x = "65" y = "150"> HTML5 + CSS3 炫酷实例集锦<animateTransform dur = "2s" attributeName =
"transform" repeatCount = "indefinite" type = "rotate" from = "0,200, 160" to = "360,200,160"/></text>
</svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<animateTransform dur="2s" attributeName="transform" repeatCount="indefinite" type="rotate" from="0, 200,160" to="360,200,160"/>表示在两秒的时间内以(200, 160)点为中心旋转 360°，且永不停歇。如果需要将文字放大两倍，则应该将上述代码修改为<animateTransform dur="2s" attributeName="transform" repeatCount="indefinite" type="scale" from="1" to="2"/>。

此实例的源文件名是 myHtmlA223.html。

023　在 SVG 中逐字旋转一行的每个文字

此实例主要在 SVG 中设置<text>标签的 rotate 属性，从而实现<text>标签中的每个汉字都以不同的角度进行旋转。当在 Google Chrome 浏览器中显示该页面时，"炫酷实例集锦"中的每个字旋转之后的效果如图 023-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  .mySVG { width: 400px;height: 400px;}
  .myText { font-size: 48px;font-weight: bold;Letter-spacing:30px; }
```

```
</style></head>
<body><div><svg class = "mySVG">
  <text x = "0" y = "60" rotate = "30,60,90,120,150,180" class = "myText">炫酷实例集锦</text></svg></
div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<text x="0" y="60" rotate="30,60,90,120,150,180" class="myText">炫酷实例集锦</text>中的 rotate 属性值"30,60,90,120,150,180"表示 6 个角度,分别用于旋转"炫酷实例集锦"这 6 个汉字。如果汉字的数量多于角度的数量,则以最后一个角度作为剩余汉字的旋转角度,例如<text x="0" y="60" rotate="30,60,90,120,150,180" class="myText">炫酷实例集锦老虎实作</text>中的"老虎实作"无法对应 rotate 属性值,则以 rotate 属性值的最后一个角度 180°作为其旋转角度。

此实例的源文件名是 myHtmlA226.html。



图 023-1

024 在 SVG 中压扁单行文本的每个文字

此实例主要在 SVG 中设置<text>标签的 textLength 和 lengthAdjust 属性,从而实现<text>标签中的每个汉字都水平拉伸,以扁平的风格显示。当在 Google Chrome 浏览器中显示该页面时,"炫酷实例集锦"中的每个字以扁平风格显示,效果如图 024-1 所示。有关此实例的主要代码如下。

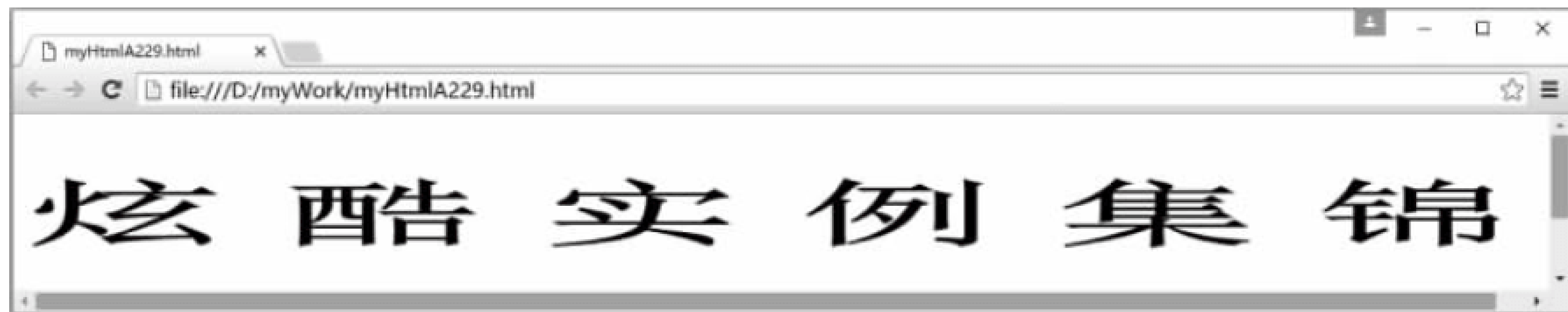


图 024-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .mySVG { width: 1200px; height: 200px; }
  .myText { font - size: 56px; font - weight: bold; letter - spacing: 15px;}
 </style></head>
<body><div><svg class = "mySVG">
  <text x = "0" y = "90" textLength = "1150" lengthAdjust = "spacingAndGlyphs" class = "myText">
炫酷实例集锦</text></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,< text x="0" y="90" textLength="1150" lengthAdjust="spacingAndGlyphs" class="myText">炫酷实例集锦</text >表示在宽度为1150px 的范围内以矢量图形的方式平均排列“炫酷实例集锦”,由于“炫酷实例集锦”的默认宽度远远小于 1150px,因此一旦执行拉伸操作,则出现扁平化的效果。

此实例的源文件名是 myHtmlA229. html。

025 在 SVG 中实现单行文本的字间距不等

此实例主要在 SVG 中设置< text >标签的 dx 属性,从而实现< text >标签中的每个汉字都有不同的字间距。当在 Google Chrome 浏览器中显示该页面时,“炫酷实例集锦”中每个字的间距都不相同,效果如图 025-1 所示。有关此实例的主要代码如下。



图 025-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .mySVG { width: 450px; height: 400px;}
  .myText { font - size: 56px; font - weight: bold;}
 </style></head>
<body><div><svg class = "mySVG">
  <text x = "0" y = "100" dx = "0,5,15,20,25,30" class = "myText">炫酷实例集锦</text>
 </svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,< text x="0" y="100" dx="0,5,15,20,25,30" class="myText">炫酷实例集锦</text >中的 dx 属性值"0,5,15,20,25,30"表示 6 个间距,分别用于对应“炫酷实例集锦”的每个汉字,类似于每个汉字的 margin-left。

此实例的源文件名是 myHtmlA227. html。

026 在 SVG 中错落显示单行文本的每个文字

此实例主要在 SVG 中设置< text >标签的 y 属性,从而实现< text >标签中的每个汉字都有不同的垂直坐标,以高低不同的错落风格显示。当在 Google Chrome 浏览器中显示该页面时,“炫酷实例集锦”中的每个字错落显示效果如图 026-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .mySVG { width: 450px; height: 200px;}
```

```
  /*设置字体的基本样式*/
  .myText { font-size: 56px; font-weight: bold;letter-spacing: 15px;}
 </style></head>
<body><div><svg class="mySVG"><text x="0" y="60,99,70,55,70,90" class="myText">炫酷实例集锦
</text></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<text x="0" y="60,99,70,55,70,90" class="myText">炫酷实例集锦</text>中的 y 属性值"60,99,70,55,70,90"表示 6 个垂直坐标，分别用于对应"炫酷实例集锦"这 6 个汉字，类似于每个汉字的 y 值。如果汉字的数量多于垂直坐标的数量，则以最后一个垂直坐标作为剩余汉字的垂直坐标，例如<text x="0" y="60,99,70,55,70,90" class="myText">炫酷实例集锦老虎实作</text>中的"老虎实作"无法对应 y 属性值，则以 y 属性值的最后一个垂直坐标 90 作为其垂直坐标。

此实例的源文件名是 myHtmlA228.html。



图 026-1

027 以阴影模糊效果显示当前选择文本

此实例主要设置 text-shadow 属性值，从而实现以阴影模糊效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以阴影模糊的特效显示，如图 027-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以阴影模糊的特效显示。有关此实例的主要代码如下。



图 027-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*当鼠标指针悬停在书名上时显示阴影特效文字*/
```

```
    p:hover {text-shadow: 5px 5px 5px gray; color: navy;
             font-size: 40px; font-weight: bold; font-family: 宋体;}
    p{padding-left: 8px;padding-top: 8px;padding-bottom: 8px;
      width: 380px; margin: 10px; background-color: lightcyan; border-radius: 5px;}
  </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果，它是使用 CSS3 属性增加文字的质感而不使用任何图片。该属性是以逗号分隔的阴影列表，每个阴影由两个或 3 个长度值和一个可选的颜色值进行规定，省略的长度是 0。text-shadow 属性的设置语法如下。

```
text-shadow: h-shadow v-shadow blur color;
```

其中，h-shadow 表示水平阴影的位置，允许为负值；v-shadow 表示垂直阴影的位置，允许为负值；blur 可选，表示模糊的距离；color 可选，表示阴影的颜色。

此实例的源文件名是 myHtmlB034.html。

028 以霓虹灯发散效果显示当前选择文本

此实例主要设置 text-shadow 属性值，从而实现以霓虹灯发散效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以霓虹灯发散的特效显示，如图 028-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以霓虹灯发散的特效显示。有关此实例的主要代码如下。



图 028-1

```
<!doctype html><html><head><meta charset="UTF-8">
<style type="text/css">
  /* 当鼠标指针悬停在书名上时显示霓虹灯效果的文字 */
  p:hover { text-shadow: 0 0 5px #FF0000; color: black;
            font-size: 40px;font-weight: bold; font-family: 宋体;}
  p {padding-left: 8px;padding-top: 8px;padding-bottom: 8px;
     width: 380px; margin: 10px;
     background-color: lightcyan;border-radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

当需要使文字产生霓虹灯发散效果时应结合字体的大小设置 blur 值。

此实例的源文件名是 myHtmlB035.html。

029　以多级辉光效果显示当前选择文本

此实例主要设置 text-shadow 的多级属性值，从而实现以多级辉光效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以多级辉光的特效显示，如图 029-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以多级辉光的特效显示。有关此实例的主要代码如下。



图　029-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  /* 当鼠标指针悬停在书名上时显示多级辉光效果的文字 */
  p:hover { text - shadow: 0 0 5px #FFF, 0 0 10px #FFF, 0 0 15px #FFF, 0 0 30px #FF00DE, 0 0 45px #
FF00DE; color: black; font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica,
Arial, Sans;}
  p { padding - left: 8px;padding - top: 8px;padding - bottom: 8px;width: 380px; margin: 10px;background
 - color: lightcyan;border - radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中，blur 表示模糊的距离。通常，设置比较大的模糊距离可以增加辉光效果，当然也可以像实例这样同时增加几个不同的距离值，从而创造多种不同的阴影效果。

此实例的源文件名是 myHtmlB036.html。

030　以下沉凹坑效果显示当前选择文本

此实例主要在 text-shadow 属性值中仅设置垂直方向上的阴影值，从而实现以下沉凹坑效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以下沉凹坑的特效显示，如图 030-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以下沉凹坑的特效显示。有关此实例的主要代码如下。



图　030-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*当鼠标指针悬停在书名上时显示下沉凹坑效果的文字*/
  p:hover { text-shadow: 0 1px 1px #FFF; color: black;
            font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans;}
  p {padding-left: 8px;padding-top: 8px;padding-bottom: 12px;
     width: 380px; margin: 10px;background-color: gray; border-radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text-shadow: h-shadow v-shadow blur color;
```

其中，h-shadow 表示水平阴影的位置，允许为负值；v-shadow 表示垂直阴影的位置，允许为负值。在此实例(text-shadow：0 1px 1px #FFF)中，水平方向的阴影值为 0，垂直方向的阴影值为 0，因此产生了这种特殊的下沉凹坑的阴影效果。用户也可以设置垂直方向的阴影值为 0，例如“text-shadow：1px 0　1px #FFF”，从而产生另外的阴影效果。

此实例的源文件名是 myHtmlB037.html。

031　以雕刻凸出效果显示当前选择文本

此实例主要在 text-shadow 属性值中设置垂直方向和水平方向的阴影值为负值，从而实现以雕刻凸出效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以雕刻凸出的特效显示，如图 031-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以雕刻凸出的特效显示。有关此实例的主要代码如下。



图　031-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*当鼠标指针悬停在书名上时显示雕刻凸出效果的文字*/
  p:hover {text-shadow: -1px -1px 0 #FFF,1px 1px 0 #333,1px 1px 0 #444;
          color:black; font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica,
          Arial, Sans;}
  p {padding-left: 8px;padding-top: 8px;padding-bottom: 12px;width: 380px;
     margin: 10px;background-color: gray;border-radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text-shadow: h-shadow v-shadow blur color;
```

其中，h-shadow 表示水平阴影的位置，允许为负值；v-shadow 表示垂直阴影的位置，允许为负值。在此实例(text-shadow：－1px －1px 0 ＃FFF，1px 1px 0 ＃333，1px 1px 0 ＃444)中，第一级的水平方向的阴影值为－1，垂直方向的阴影值为－1；第二、三级的水平方向的阴影值为 1，垂直方向的阴影值为 1，因此产生了这种特殊的雕刻凸出的阴影效果。

此实例的源文件名是 myHtmlB038.html。

032　以模糊发散效果显示当前选择文本

此实例主要在 text-shadow 属性值中仅设置模糊距离值 blur，并设置 color 属性值为 transparent，从而实现以模糊发散效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以模糊发散的特效显示，如图 032-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以模糊发散的特效显示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   /*当鼠标指针悬停在书名上时显示模糊发散效果的文字*/
   p:hover {text-shadow: 0 0 5px #F96; color: transparent;
```

```
          font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans;}
   p {padding-left: 8px; padding-top: 8px; padding-bottom: 12px; width: 380px;
     margin: 10px; background-color: gray;border-radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text-shadow: h-shadow v-shadow blur color;
```

其中，blur 可选，表示模糊的距离；color 可选，表示阴影的颜色。在使用 text-shadow 制作模糊发散文字效果时需要注意设置文本的前景色 color 为 transparent，模糊距离值 blur 越大，其效果越模糊；其二不设置水平方向和垂直方向的偏移值。

此实例的源文件名是 myHtmlB039.html。

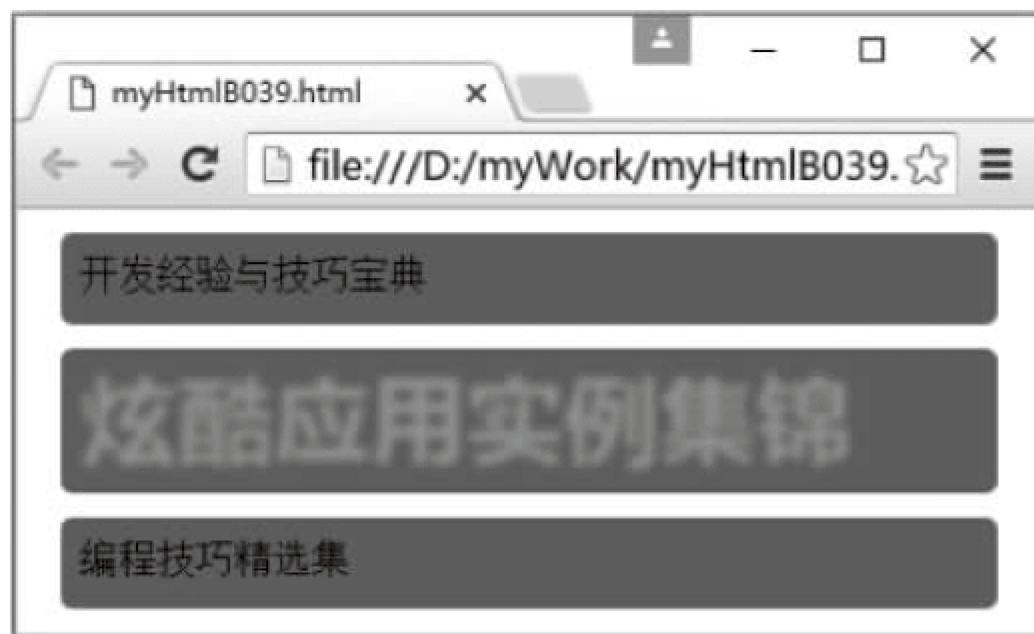


图 032-1

033 以模糊雕刻效果显示当前选择文本

此实例主要在 text-shadow 属性值中设置多级阴影值和模糊值，并设置 color 属性值为 transparent，从而实现以模糊雕刻效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以模糊雕刻的特效显示，如图 033-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以模糊雕刻的特效显示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /* 当鼠标指针悬停在书名上时显示模糊雕刻效果的文字 */
  p:hover { text-shadow:0 0 6px #F96, -1px -1px #FFF, 1px -1px #444;
          color: transparent; font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans",
          Helvetica, Arial, Sans;}
  p {padding-left: 8px; padding-top: 8px; padding-bottom: 12px; width: 380px;
     margin: 10px;background-color: gray;border-radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```



图　033-1

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中，blur 可选，表示模糊的距离；color 可选，表示阴影的颜色。在使用 text-shadow 制作模糊雕刻文字效果时需要注意设置文本的前景色 color 为 transparent，模糊距离值 blur 越大，其效果越模糊；其二设置水平方向和垂直方向的偏移值以产生雕刻效果。

此实例的源文件名是 myHtmlB040. html。

034　以内凹嵌入效果显示当前选择文本

此实例主要在 text-shadow 属性值中设置阴影值和模糊值，并设置 color 属性值、background 属性值以及阴影颜色值，从而实现以内凹嵌入效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以内凹嵌入的特效显示，如图 034-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以内凹嵌入的特效显示。有关此实例的主要代码如下。



图　034-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  /* 当鼠标指针悬停在书名上时显示内凹嵌入效果的文字 */
  p:hover { text - shadow: 1px 1px 0 #E4F1FF; color: #566F89; font: bold 40px/100% "微软雅黑", "Lucida
          Grande", "Lucida Sans", Helvetica, Arial, Sans;}
```

```
  p {padding - left: 8px; padding - top: 8px; padding - bottom: 12px; width: 380px;
     margin: 10px; background: lightblue; border - radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中，blur 可选，表示模糊的距离；color 可选，表示阴影的颜色。在使用 text-shadow 制作内凹嵌入文字效果时需要注意设置文本的前景色 color、背景色 background 以及阴影色 color，这 3 种颜色需要合理搭配，模糊距离值 blur 通常为 0；其二设置水平方向的偏移值和垂直方向的偏移值以产生嵌入效果。这种效果应该文字的前景色比背景色暗，阴影颜色稍微比背景色亮，这一点很重要，如果阴影色太亮看起来会怪，如果太暗将没有效果显示。

此实例的源文件名是 myHtmlB041.html。

035　以线条描边效果显示当前选择文本

此实例主要在 text-shadow 属性中设置阴影值和阴影颜色值，从而在文字线条的外围实现线条描边的效果。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以线条描边的特效显示，如图 035-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以线条描边的特效显示。有关此实例的主要代码如下。



图　035-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
   /* 当鼠标指针悬停在书名上时显示线条描边的特效文字 */
   p:hover {text - shadow: 1px 1px 0px lightgray, - 1px  - 1px 0px lightgray;color: snow;
            font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans;}
   p { padding - left: 8px;padding - top: 8px;padding - bottom: 12px;
       width: 380px;margin: 10px; background: lightpink;border - radius: 5px;}
 </style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴

影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中,h-shadow 表示水平阴影的位置,允许为负值;v-shadow 表示垂直阴影的位置,允许为负值;blur 可选,表示模糊的距离;color 可选,表示阴影的颜色。此实例在使用 text-shadow 属性实现线条描边的文字特效时主要运用两个阴影,第一个是向左上投影(1px 1px 0px lightgray),第二个是向右下投影(－1px －1px 0px lightgray),用户还需注意制作文字描边的阴影效果不使用模糊距离值。

此实例的源文件名是 myHtmlB042.html。

036　以浮雕镶嵌效果显示当前选择文本

此实例主要在 text-shadow 属性中设置多个阴影,从而实现以浮雕镶嵌效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在“火树银花合”上,则出现如图 036-1 所示的效果;如果将鼠标指针悬浮在“星桥铁锁开”或“灯树千光照”上,也会出现类似的效果。有关此实例的主要代码如下。



图　036-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  /* 当鼠标指针悬停时显示浮雕镶嵌效果的文字 */
  p:hover {color: #FFF; font - size: 80px;
           text - shadow: 0 1px 0 #CCC, 0 2px 0 #C9C9C9, 0 3px 0 #BBB, 0 4px 0 #B9B9B9, 0 5px 0 #AAA, 0
           6px 1px rgba(0,0,0,0.1), 0 0 5px rgba(0,0,0,0.1),0 1px 3px rgba(0,0,0,0.3),0 3px 5px rgba(0,
           0,0,0.2),0 5px 10px rgba(0,0,0,0.25);}
  p { padding - left: 8px; padding - top: 8px; padding - bottom: 8px; width: 400px; margin: 10px;
      background - color:lightseagreen;border - radius: 5px;}
 </style></head>
<body><p>火树银花合</p><p>星桥铁锁开</p><p>灯树千光照</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中,h-shadow 表示水平阴影的位置,允许为负值;v-shadow 表示垂直阴影的位置,允许为负

值;blur 可选,表示模糊的距离;color 可选,表示阴影的颜色。此实例中的浮雕镶嵌文字特效主要是通过该属性以多个阴影叠加,并且通过设置透明度产生的。

此实例的源文件名是 myHtmlB110.html。

037 以渐变倒影效果显示当前选择文本

此实例主要设置-webkit-box-reflect 属性,从而实现以渐变倒影效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在文本上,则将出现该文本的倒影,如图 037-1 所示。有关此实例的主要代码如下。



图 037-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /*绘制渐变色文字*/
  div { font-size: 80px; font-weight: bold; cursor:pointer;
       background-image: -webkit-gradient(linear,left top,left bottom, from
       (lightgreen), to(blue)); -webkit-background-clip: text;
       -webkit-text-fill-color: transparent;}
   /*在鼠标指针悬浮时产生倒影*/
  div:hover{/*倒影有线性渐变效果*/
          -webkit-box-reflect:below 0px linear-gradient(transparent 0%,white);
          /*倒影无渐变效果*/
          /* -webkit-box-reflect:below 0px; */}
 </style></head>
<body><div>炫酷实例集锦</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,div:hover{-webkit-box-reflect:below 0px linear-gradient(transparent 0%,white);}用于实现当鼠标指针悬浮在文本上时,在文本下方的 0px 间隔距离处添加渐变的倒影文本。此外,如果设置-webkit-box-reflect 的属性值分别为 above、left、right,则可以在文本的上方、左边、右边产生倒影。

此实例的源文件名是 myHtmlB191.html。

038 以透明阴影效果显示当前选择文本

此实例主要设置 text-shadow 属性并对文本使用完全透明的颜色,从而实现以透明阴影效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时,"春花秋月何时了"将以透明阴影效

果显示，如图 038-1 所示；如果将鼠标指针悬浮在这行透明文本上，则文本将以正常状态显示。有关此实例的主要代码如下。

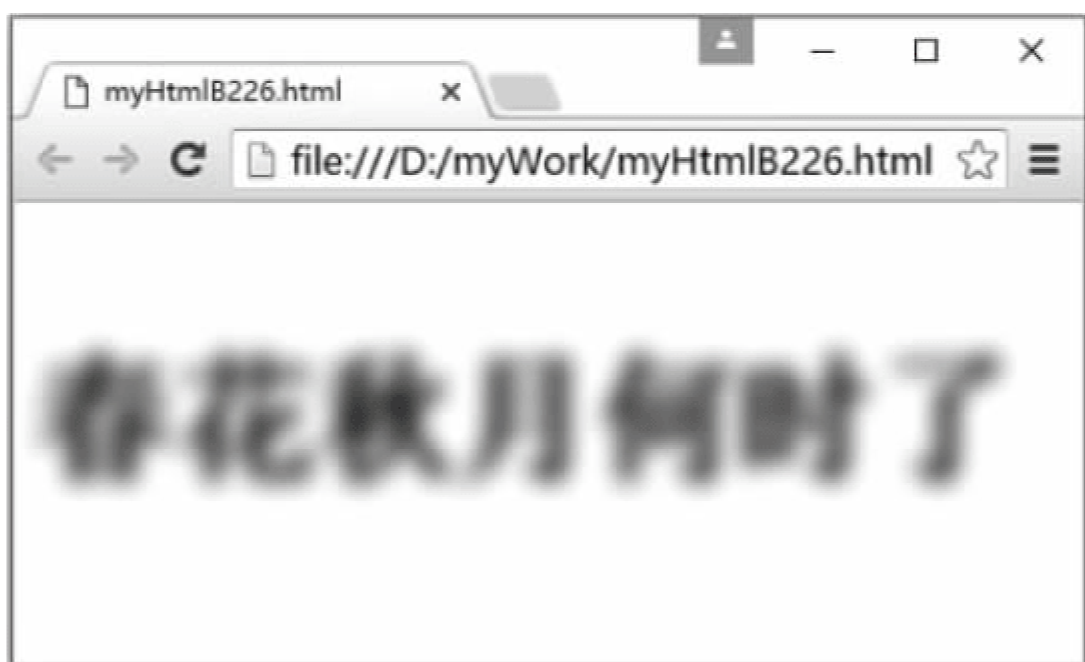


图　038-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  p { font-size: 60px; font-weight: bold;
      text-shadow: 0 0 15px #000;          /* 添加文字阴影 */
      color: rgba(0, 0, 0, 0);             /* 设置颜色为透明色 */}
  p:hover { text-shadow: none;             /* 取消文字阴影 */
            color: #000;                   /* 设置颜色为黑色 */}
 </style></head>
<body><p>春花秋月何时了</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow：0 0 15px #000 用于给文字加上阴影效果，它是使用 CSS3 属性增加文字的质感而不使用任何图片。color：rgba(0，0，0，0)用于设置文字颜色为透明色，在默认情况下，它使文本不可见，但在此实例中由于文字有阴影，用户正好可以看见文字阴影的那种模糊状态。

此实例的源文件名是 myHtmlB226.html。

039　设置文字边框创建镂空风格的文字

此实例主要通过设置元素的 text-stroke-width 属性和 text-stroke-color 属性创建镂空风格的文字。当在 Google Chrome 浏览器中显示该页面时，“炫酷实例”将以镂空风格显示，如图 039-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
<style type="text/css">
   p { color: transparent; -webkit-text-stroke-width:1px;
        -webkit-text-stroke-color:green; font-size: 100px; margin: 30px;}
 </style></head>
<body><p>炫酷实例</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-text-stroke-width：1px 表示每个文字的线条边框宽度是 1px，-webkit-text-stroke-color：green 表示每个文字的线条边框颜色是绿色，这两个属性可以合并到-webkit-text-stroke-width 属性中表示。



图 039-1

此实例的源文件名是 myHtmlB145.html。

040 使用多级阴影创建 3D 效果的文字

此实例主要通过在 text-shadow 属性中设置阴影值和阴影颜色值实现以 3D 立体效果显示当前选择的文本。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在第二本书名上，则该书名将以 3D 立体的特效显示，如图 040-1 所示；如果将鼠标指针悬浮在第一本或第三本书名上，则该书名也将以 3D 立体的特效显示。有关此实例的主要代码如下。



图 040-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /* 当鼠标指针悬停在书名上时显示 3D 立体的特效文字 */
  p:hover {text-shadow: 1px 1px rgba(197, 223, 248, 0.8), 2px 2px rgba(197, 223, 248, 0.8), 3px 3px rgba
  (197, 223, 248, 0.8), 4px 4px rgba(197, 223, 248, 0.8), 5px 5px rgba(197, 223, 248, 0.8), 6px 6px rgba
  (197, 223, 248, 0.8);
           color: #FFF; font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica,
           Arial, Sans;}
  /* 当鼠标指针悬停在书名上时显示 3D 立体的特效文字,效果二; 自动覆盖效果一 */
  p:hover {text-shadow: -1px -1px rgba(197, 223, 248,0.8), -2px -2px rgba(197, 223, 248,0.8), -3px
           -3px rgba(197, 223, 248,0.8), -4px -4px rgba(197, 223, 248,0.8), -5px -5px rgba(197,
           223, 248,0.8), -6px -6px rgba(197, 223, 248,0.8);
           color: #FFF; font: bold 40px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica,
           Arial, Sans;}
  p {padding-left: 8px;padding-top: 8px;padding-bottom: 12px;width: 380px;
     margin: 10px; background: grey;border-radius: 5px;}
```

```
</style></head>
<body><p>开发经验与技巧宝典</p><p>炫酷应用实例集锦</p><p>编程技巧精选集</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text-shadow: h-shadow v-shadow blur color;
```

其中，h-shadow 表示水平阴影的位置，允许为负值；v-shadow 表示垂直阴影的位置，允许为负值；blur 可选，表示模糊的距离；color 可选，表示阴影的颜色。

3D 文字效果的运用原理如同 Photoshop，在文字的下方或上方复制多个图层，并把每一个层向左上或右下方向移动 1px 距离，从而制作出 3D 效果，层数越多，效果越厚重。如果换成用 text-shadow 制作就是使用多个阴影，并把阴影色设置相同，使用 rgba 透明色效果更佳，例如实例“text-shadow：−1px −1px rgba(197，223，248,0.8)，−2px −2px rgba(197，223，248,0.8)，−3px −3px rgba(197，223，248,0.8)，−4px −4px rgba(197，223，248,0.8)，−5px −5px rgba(197，223，248,0.8)，−6px −6px rgba(197，223，248,0.8)”中的 0.8 即表示透明度值。

此实例的源文件名是 myHtmlB043.html。

041　裁剪圆形并使文字环绕圆形边缘显示

此实例主要通过设置元素的 shape-outside 属性和使用 before 选择器实现裁剪圆形图片并使文字环绕圆形边缘显示。当在 Google Chrome 浏览器中显示该页面时，右端的文字将沿着左端的圆形图片边缘换行显示，如图 041-1 所示。有关此实例的主要代码如下。



图　041-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
```

```
        * { margin: 0px; padding: 0px; position: relative;box-sizing: border-box;}
            .content {padding: 10px;text-align: justify;color: black;font-size: 15px;}
            .content::before {content: ""; display: block; float: left;
                              width: 360px; height: 350px; shape-outside: circle(50%); }
       p { margin: 10px;text-indent: 1px; font-size: 13px;}
       .myDiv {width: 385px;height: 356px; border-radius: 5px; position: absolute;
               background-image: url(img/B115.jpg); text-align: center;
               -webkit-clip-path: circle(45% at 50% 50%); }
        body { background-color: aquamarine; }
    </style>
  </head>
  <body><div class="myDiv"></div><div class="content">
    <p>武当山,道教圣地,位于湖北省十堰市武当山旅游经济特区。武当山又名太和山、谢罗山、参上山、仙室
山,古有"太岳"、"玄岳"、"大岳"之称。截至2013年,武当山有古建筑53处,建筑面积2.7万平方米,建筑遗址9
处,占地面积20多万平方米,全山保存各类文物5035件。明代,武当山被皇帝封为"大岳"、"治世玄岳",被尊为
至高无上的"皇室家庙"。武当山以"四大名山皆拱揖,五方仙岳共朝宗"的"五岳之冠"的显赫地位闻名于世。武
当山是联合国公布的世界文化遗产地之一,是中国国家重点风景名胜区、国家AAAAA级风景区。武当山也是道
教名山和武当武术的发源地,被称为"亘古无双胜境,天下第一仙山"。</p>
  </div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,myDiv类裁剪的圆形图片仅用于显示,不是导致右端圆弧文字换行的直接原因,在content::before中绘制的圆形才是导致右端圆弧文字换行的直接原因,它不可见。在此实例中将上述两个圆形绘制成大小差不多,因此给用户的感觉就像是右端的圆弧换行文字是由左端的圆形图片制造的这种效果。content::before表示在content的前面设置的CSS样式。

此实例的源文件名是myHtmlB115.html。

042　裁剪多边形并使文字环绕其边缘显示

此实例主要通过设置元素的shape-outside属性和使用before选择器实现以五角星(多边形)形状裁剪图片并使文字环绕其边缘显示。当在Google Chrome浏览器中显示该页面时,左端的文字将沿着右端的五角星图片边缘换行显示,如图042-1所示。有关此实例的主要代码如下。



图　042-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   * { margin: 0px; padding: 0px; position: relative; box-sizing: border-box; }
       .content { padding: 10px; text-align: justify; color: black; font-size: 15px;}
       .content::before { display: block; float: right;width: 385px; height: 356px;
                          shape-outside: polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82%
                          100%, 50% 75%, 18% 100%, 31% 59%, 0 38%, 37% 38%);}
p { margin: 10px; text-indent: 1px; font-size: 13px; }
   .myDiv {width: 385px;height: 356px;left:60px;top:10px;border-radius: 5px;
   position: absolute; background-image: url(img/B116.jpg); text-align: center;
   -webkit-clip-path: polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82% 100%, 50% 75%,
   18% 100%, 31% 59%, 0 38%, 37% 38%); }
   body { background-color: aquamarine;}
 </style></head>
<body><div class="myDiv"></div><div class="content">
   <p>黄山,位于安徽省黄山市,原名黟山,唐朝时更名为黄山,取自"黄帝之山"之意。黄山是世界自然和文化双遗产,世界地质公园,中国十大名胜古迹之一,国家5A级旅游景区。黄山风景区面积160.6平方千米,东起黄狮,西至小岭脚,北始二龙桥,南达汤口镇,分为温泉、云谷、玉屏、北海、松谷、钓桥、浮溪、洋湖、福固九个管理区,包括200多个大小景点。黄山以奇松、怪石、云海、温泉、冬雪"五绝"著称于世,拥有"天下第一奇山"之称。"五岳归来不看山,黄山归来不看岳"是对黄山最好的评价。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,content::before 选择器的 shape-outside: polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82% 100%, 50% 75%, 18% 100%, 31% 59%, 0 38%, 37% 38%)用于绘制五角星,但此五角星是不可见的,主要用于控制文本沿五角星的边缘显示;float:right 表示多边形(五角星)在文字的右边,如果需要让多边形在文字的左边,则应为 float: left。myDiv 的-webkit-clip-path: polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82% 100%, 50% 75%, 18% 100%, 31% 59%, 0 38%, 37% 38%)才是真正按照五角星的形状裁剪图片,left:60px; top:10px 用于调整裁剪五角星图片的左上角的位置,可以根据具体情况进行调整。

此实例的源文件名是 myHtmlB116.html。

043 通过组合属性值实现加长阴影文字

此实例主要使用循环组合属性值设置 text-shadow 属性,从而实现为文字添加加长阴影的特效。当在 Google Chrome 浏览器中显示该页面时,加长阴影文字效果如图 043-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
    $(function() {
     var myShadow = "";
     for (var i = 1; i <= 5; i++) {     //使用循环组合多个阴影属性值
        myShadow += "black " + i + "px " + i + "px 0px,";
     }
     $(".myBox").css("text-shadow", myShadow); });
 </script>
 <style type="text/css">
```

```
  .myBox {/ * font - family: "Arial","Microsoft YaHei","黑体","宋体",sans - serif;; * /
          font - size: 60px; font - weight: bold; display: inline - block;
          margin - top: 20px; letter - spacing:2px; color:green; }
</style></head>
<body><div align = "center"><div class = "myBox">炫酷实例集锦</div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,text-shadow 属性用于给文字加上阴影和模糊效果。text-shadow 属性的设置语法如下。

```
text - shadow: h - shadow v - shadow blur color;
```

其中,h-shadow 表示水平阴影的位置,允许为负值;v-shadow 表示垂直阴影的位置,允许为负值;blur 可选,表示模糊的距离;color 可选,表示阴影的颜色。该实例中的 $(function () {…}…)的作用是在文字的下方复制多个 text-shadow 阴影层,并把每一层向右下方向移动 1px 距离,从而制作出 3D 效果,即设置 text-shadow: black 1px 1px 0px, black 2px 2px 0px, black 3px 3px 0px, black 4px 4px 0px, black 5px 5px 0px,产生的效果如图 043-1 所示。若直接设置 text-shadow: black 5px 5px 0px,产生的效果如图 043-2 所示,这显然不是创建立体字。

此实例的源文件名是 myHtmlB369.html。



图 043-1



图 043-2

044 创建渐变背景绘制上下渐变的文字

此实例主要通过设置 background-image、-webkit-background-clip 和-webkit-text-fill-color 等属性实现创建渐变色背景绘制由上向下渐变的文字。当在 Google Chrome 浏览器中显示该页面时,渐变色文字“炫酷实例”的效果如图 044-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
<style type = "text/css">
  .text - gradient { font - size: 150px; font - family: '微软雅黑';
     background - image: - webkit - gradient(linear,left top,left bottom, from(lightgreen), to(blue));
     - webkit - background - clip: text; - webkit - text - fill - color: transparent; }
   * { margin: 0; padding: 0;}
 </style></head>
<body><h3 class = "text - gradient">炫酷实例</h3></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-image:-webkit-gradient (linear,left top,left bottom, from(lightgreen), to(blue))用于创建上绿下蓝的渐变色背景;-webkit-



图 044-1

background-clip：text 表示使用当前元素的文本线条轮廓裁剪背景；-webkit-text-fill-color：transparent 表示当前文本颜色透明。此实例主要通过该透明策略实现在文字线条中显示背景中渐变的颜色，若同时设置 text-fill-color 和 color，text-fill-color 定义的颜色将覆盖 color 属性。

此实例的源文件名是 myHtmlB186.html。

045 创建透明层绘制上下渐变的文字

此实例主要通过设置-webkit-mask-image 属性创建渐变透明层，从而绘制由上向下渐变的文字。当在 Google Chrome 浏览器中显示该页面时，渐变色文字"炫酷实例集锦"的效果如图 045-1 所示。有关此实例的主要代码如下。



图 045-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <style type = "text/css">
   .text - gradient { font - size: 5em; font - weight: bold;
                      position: absolute; color:green;overflow - x:hidden;width:600px; }
   .text - gradient[myData]::after { content: attr(myData); color:red;
            /* 在绿色"炫酷实例集锦"上叠加红色"炫酷实例集锦" */
            position: absolute;left: 0;
            /* 通过渐变透明层控制叠加效果 */
             - webkit - mask - image: - webkit - gradient(linear, 0 top, 0 bottom, from(rgba(255,255,255,
1)), to(rgba(255,255,255, 0))); }
</style></head>
<body><div class = "text - gradient" myData = "炫酷实例集锦">炫酷实例集锦</div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-mask-image：-webkit-

gradient(linear, 0 top, 0 bottom, from(rgba(255,255,255,1)), to(rgba(255, 255,255, 0)))用于创建渐变透明层,其中的 from(rgba(255,255,255,1))和 to(rgba(255, 255,255, 0)))的颜色值无关紧要,关键是 rgba 的第 4 个参数 1 和 0,该参数代表透明度,0 表示完全透明,1 表示完全不透明。

此实例的源文件名是 myHtmlB187. html。

046 将图片颜色和文本颜色混合叠加显示

此实例主要设置元素的 mixBlendMode 属性值为 overlay,从而实现将图片的颜色和文本的颜色混合叠加显示。当在 Google Chrome 浏览器中显示该页面时,单击"叠加模式"按钮则图片的颜色和文本的颜色将混合叠加在一起,如图 046-1 所示;单击其他两个按钮将实现按钮标题所示的功能。有关此实例的主要代码如下。



图 046-1

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnOverlay").click(function() {       //叠加模式
    $("h1").css("mixBlendMode", "overlay");
   });
   $("#myBtnHard").click(function() {          //强光模式
    $("h1").css("mixBlendMode", "hard-light");
   });
   $("#myBtnSoft").click(function() {          //柔光模式
    $("h1").css("mixBlendMode", "soft-light"); });});
 </script>
 <style>
  div {width: 400px;height: 250px; border-radius: 5px; justify-content: center;
       background-image: url(img/B113.jpg);display: flex; align-items: center;
       background-repeat: no-repeat; background-size: 100% 100%; margin: 3px;}
  h1 {color: yellow;font-size:60px;}
  input { width: 120px; border-radius: 5px; margin: 5px;}
 </style></head>
<body><input type="button" value="叠加模式" id="myBtnOverlay"/>
<input type="button" value="强光模式" id="myBtnHard"/>
```

```
<input type="button" value="柔光模式" id="myBtnSoft"/>
<div><h1>重庆园博园</h1></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("h1"). css("mixBlendMode"," overlay")用于设置"重庆园博园"文本的颜色与底层图片的颜色进行叠加显示。除了 overlay 以外,mixBlendMode 属性还支持属性值 normal(正常)、multiply(正片叠底)、screen(滤色)、darken(变暗)、lighten(变亮)、color-dodge(颜色减淡)、color-burn(颜色加深)、hard-light(强光)、soft-light(柔光)、difference(差值)、exclusion(排除)、hue(色相)、saturation(饱和度)、color(颜色)、luminosity(亮度)、initial(初始)、inherit(继承)、unset(复原)。

此实例的源文件名是 myHtmlB113. html。

047 使用自定义字体显示手写文字

此实例主要通过使用自定义字体规则@font-face 提取自定义字体库文件中的字体显示文本。当在 Google Chrome 浏览器中显示该页面时,使用自定义字体显示的唐诗如图 047-1 所示。有关此实例的主要代码如下。

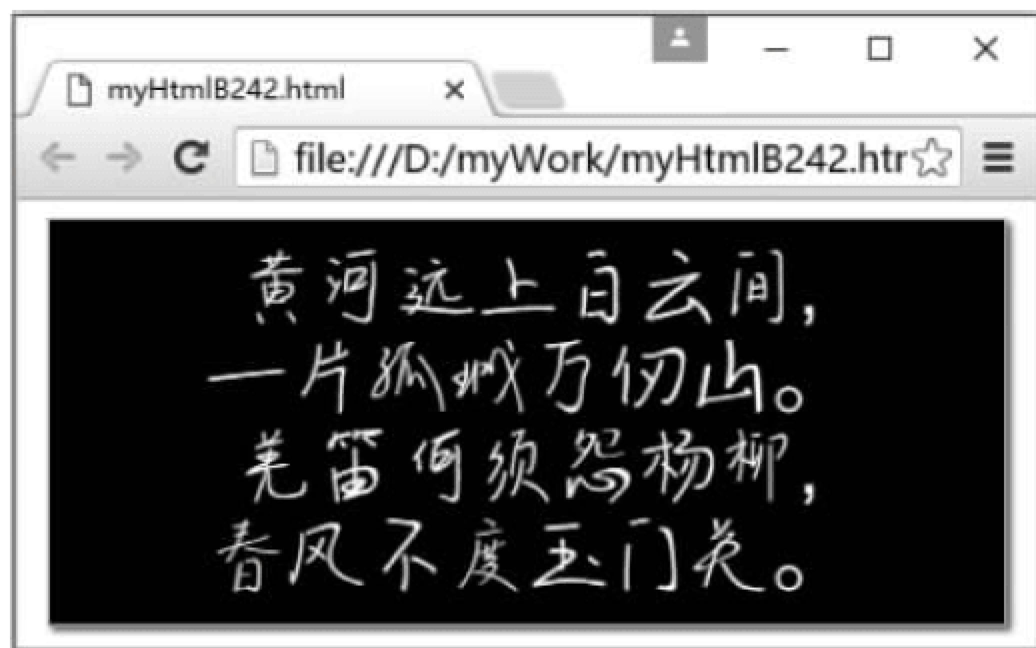


图 047-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   @font-face { font-family: cus_font; src: url('Fonts/cus_font.otf'); }
   p { margin: 5px; text-align: center;
       width: 395px; padding: 10px;
       border: 1px solid #BFBFBF; background-color: black;
       color: white; box-shadow: 2px 2px 3px gray; font-size: 35px;
        -webkit-writing-mode: horizontal-tb;font-family: cus_font;}
 </style></head>
<body><center><p>黄河远上白云间,<br>一片孤城万仞山。<br>
羌笛何须怨杨柳,<br>春风不度玉门关。</p></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,@font-face 是 CSS3 中的一个模块,其主要作用是把定义的字体嵌入到网页中。@font-face 的语法如下。

```
@font-face{font-properties}
```

其中,font-properties 的相关属性及其属性值的说明如表 047-1 所示。

表 047-1 font-properties 的相关属性及说明

字体属性	属性值	说明
font-family	name	必需的，定义字体的名称
src	URL	必需的，定义该字体下载的网址
font-stretch	normal	可选，定义该字体如何被拉长，默认值是“正常”
	condensed	
	ultra-condensed	
	extra-condensed	
	semi-condensed	
	expanded	
	semi-expanded	
	extra-expanded	
	ultra-expanded	
font-style	normal	可选，定义该字体样式，默认值是“正常”
	italic	
	oblique	
font-weight	normal	可选，定义字体的粗细，默认值是“正常”
	bold	
	100	
	200	
	300	
	400	
	500	
	600	
	700	
	800	
	900	
unicode-range	unicode-range	可选，定义该字体支持 Unicode 字符的范围，默认值是“ü+0—10 FFFF”

实际上，除表 047-1 之外的大多数常用字体属性均支持，例如 font-size。

此实例的源文件名是 myHtmlB242.html。

048 设置边框线高仿字库中的带圈文字

此实例主要通过设置元素的 border-radius 属性为 50%，从而实现高仿字库中的带圈文字特效。当在 Google Chrome 浏览器中显示该页面时，所有文字都被圈在圆形里面，如图 048-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {       //将所有文字逐字拆解成 span
   var myText = $('p').html();
   var myLetter = [];
   for (var i = 0; i < myText.length; i++) {
     myLetter[i] = '<span>' + myText[i] + '</span>'; }
   $('p').html(myLetter); });
 </script>
```

```
<style type = "text/css">
  / * 设置文字显示区域 * /
  p{ width:300px; margin: 30px auto; line - height: 40px; }
  / * 设置带圈文字样式 * /
  p span { margin: 3px; padding: 3px;border - radius: 50 % ; border: 1px solid gray;
       background - color: black; color: white;text - shadow: 2px 2px 5px gray; }
  / * 创建有阴影的盒子 * /
  div { border - radius: 10px; margin: 15px; font - family: 华文楷体;
   font - weight: bold;font - size: 24px; text - align: center;
   width: 365px; padding: 10px; border: 1px solid ＃BFBFBF;
   background - color: snow; box - shadow: 2px 2px 3px gray;
   - webkit - writing - mode: horizontal - tb; }
  body { background - color: lightgray; }
</style></head>
<body><center><div><p>故园东望路漫漫双袖龙钟泪不干马上相逢无纸笔凭君传语报平安</p></div>
</center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-radius：50％表示元素（span）四角的圆角半径均为宽高的50％，即此元素由矩形变为圆形（或椭圆形）。在CSS3中，border-radius属性是一个简写属性，用于设置4个border-＊-radius属性，该属性的语法格式如下。

```
border-radius: 1 - 4 length| % /1 - 4 length| % ;
```

如果以length表示，则以宽度或高度的像素值为大小；如果以％表示，则以宽度或高度的像素值的百分比为大小。此外，如果省略bottom-left（左下角），则与top-right（右上角）相同；如果省略bottom-right（右下角），则与top-left（左上角）相同；如果省略top-right（右上角），则与top-left（左上角）相同。

此实例的源文件名是myHtmlB258.html。



图　048-1

049　使用自定义字体模拟LED文字风格

此实例主要将自定义字体规则@font-face导入LED字库，从而实现以LED的风格显示文字。当在Google Chrome浏览器中显示该页面时，单击“以LED样式显示文字”按钮将以LED的风格显

示“HTML5 AND CSS3”,如图 049-1 所示;单击“以默认样式显示文字”按钮将以默认风格显示“HTML5 AND CSS3”。有关此实例的主要代码如下。



图 049-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#ledStyle").click(function(){         //以 LED 样式显示文字
    $("div h2").css("font-family","led");
   });
   $("#defaultStyle").click(function(){     //以默认样式显示文字
     $("div h2").css("font-family","initial"); });});
 </script>
 <style type="text/css">
  /* 导入自定义字体 myLed.TTF */
  @font-face { font-family: led; src: url(fonts/myLed.TTF); }
  /* 设置文字块基本样式 */
  div h2{ font-family: led; font-size: 60px;}
  button{ margin-top: 20px; width:190px; }
 </style></head>
<body><div align="center">
    <button id="ledStyle">以 LED 样式显示文字</button>
    <button id="defaultStyle">以默认样式显示文字</button><br>
<h2>HTML5 AND CSS3</h2></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,@font-face 是 CSS3 中的一个模块,其主要作用是把用户自定义的字体嵌入到网页中;font-family: led 表示使用 led 字体显示文字,注意,led 字体在自定义字库文件 myLed.TTF 中,在使用之前必须使用@font-face 导入。

此实例的源文件名是 myHtmlB348.html。

050 在二维平面中旋转单行的白色阴影文字

此实例主要通过综合使用 text-shadow 和 transform 属性实现在二维平面中旋转一行白色阴影特效文字。当在 Google Chrome 浏览器中显示该页面时,3D 阴影效果的“莫愁前路无知己”将逆时针旋转 10°,如图 050-1 所示。有关此实例的主要代码如下。



图　050-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  h1 { color: white; text-shadow:3px 3px 6px #000000; font-size: 48px;
       /* 反方向旋转 10° */
       transform: rotate(-10deg); }
  div { width:450px; height:250px; line-height: 250px;
        background-image: url(img/B240.jpg);background-size: 100% 100%; }
   * { margin: 0; overflow: hidden; }
</style></head>
<body><center><div><h1>莫愁前路无知己</h1></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow:3px 3px 6px #000000 表示阴影文字的水平阴影和垂直阴影距离是 3px，模糊距离是 6px，阴影颜色是黑色；transform: rotate(-10deg)表示将阴影文字逆时针旋转 10°。

此实例的源文件名是 myHtmlB240.html。

051　以不同颜色显示汉字的上、下两部分

此实例主要通过设置 overflow 属性为 hidden 实现以不同的颜色显示一个汉字的上、下两部分。当在 Google Chrome 浏览器中显示该页面时，“醴陵”的上半部分以灰色显示，下半部分以蓝色显示，如图 051-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  /* 在全部显示文字时用蓝色 */
  .myText { font: 240px 宋体; color: blue; }
  /* 仅上半部显示文字时用浅灰色 */
  .myHalf { height: 150px; position: absolute; overflow: hidden; color: lightgray; }
 </style></head>
<body><div><span class="myText"><span class="myHalf">醴陵</span>醴陵</span>
 </div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，起关键作用的是 height: 150px; position: absolute;overflow: hidden，它首先以蓝色显示 240px 的“醴陵”，然后在 position: absolute



图 051-1

属性的作用之下，在该蓝色"醴陵"上叠加一个同样大小的灰色"醴陵"，由于规定灰色"醴陵"的高度只有150px，并且规定隐藏(overflow: hidden)超出150px部分的内容，这样就导致蓝色"醴陵"的下面部分露出来了，从而造成上灰下蓝的显示效果。在CSS中，overflow属性规定当内容溢出元素框(盒子)时发生的事情，overflow属性的语法格式如下。

```
overflow: visible |hidden| scroll | auto | inherit
```

其中，各属性值的说明如下。

(1) visible：默认值，表示内容不会被修剪，会呈现在元素框之外。

(2) hidden：该属性值表示内容会被修剪，并且其余内容是不可见的。

(3) scroll：该属性值表示内容会被修剪，但是浏览器显示滚动条以便查看其余的内容。

(4) auto：该属性值表示如果内容被修剪，则浏览器显示滚动条以便查看其余的内容。

(5) inherit：该属性值表示应该从父元素继承overflow属性的值。

此实例的源文件名是myHtmlB182.html。

052 以不同颜色显示汉字的左、右两部分

此实例主要通过设置overflow-x属性为hidden实现以不同的颜色显示一个汉字的左、右两部分。当在Google Chrome浏览器中显示该页面时，"國"的左半部分以灰色显示，右半部分以蓝色显示，如图052-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <style type = "text/css">
  .myText { font: 400px 宋体; color: blue; }      /* 在全部显示文字时用蓝色 */
  /* 仅左半部分显示文字时用浅灰色 */
 .myHalf { width: 200px; position: absolute;overflow - x:hidden;color: lightgray;}
</style></head>
<body><div><span class = "myText"><span class = "myHalf">國</span>國</span></div> </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，起关键作用的是width: 200px; position: absolute; overflow-x: hidden，它首先以蓝色显示400px的"國"，然后在position: absolute属性的作用之下，在该蓝色"國"上叠加一个同样大小的灰色"國"，由于规定灰色"國"的宽度只有

200px，并且规定隐藏(overflow-x:hidden)超出200px部分的内容，这样就导致蓝色“國”的右半部分露出来了，从而造成左灰右蓝的显示效果。在CSS3中，overflow-x属性规定在溢出元素内容区域时是否对内容的左、右边缘进行裁剪，overflow-x属性的语法格式如下。

```
overflow - x: visible|hidden|scroll|auto|no - display|no - content;
```

其中，各属性值的说明如下。

(1) visible：该属性值表示不裁剪内容，可能会显示在内容框之外。

(2) hidden：该属性值表示裁剪内容，不提供滚动机制。

(3) scroll：该属性值表示裁剪内容，提供滚动机制。

(4) auto：该属性值表示如果溢出内容框，则应该提供滚动机制。

(5) no-display：该属性值表示如果内容不适合内容框，则删除整个框。

(6) no-content：该属性值表示如果内容不适合内容框，则隐藏整个内容。

此实例的源文件名是myHtmlB183.html。



图　052-1

053　模拟古诗的风格从上到下显示文本

此实例主要通过设置元素的writing-mode属性实现以类似于古诗的风格从上到下显示文本。当在Google Chrome浏览器中显示该页面时，默认情况下古诗将按照水平方向从左到右显示，单击“以垂直方式从右到左显示”按钮，则古诗的显示风格如图053-1所示；单击“以垂直方式从左到右显示”按钮，则古诗的显示风格如图053-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtnVerticalRL").click(function() {       //以垂直方式从右到左显示
```

```
    $("div").css("-webkit-writing-mode", "vertical-rl");
   });
   $("#myBtnVerticalLR").click(function() {      //以垂直方式从左到右显示
    $("div").css("-webkit-writing-mode", "vertical-lr"); });});
 </script>
 <style type="text/css">
  *{ margin: 0; padding: 0; }
  div{/*创建有阴影的盒子*/
      float:left; margin:5px; font-family: 华文楷体;font-size:24px;
      text-align: center; width:395px;padding:10px;border:1px solid #BFBFBF;
      background-color:lightblue;box-shadow:2px 2px 3px gray;
      -webkit-writing-mode:horizontal-tb;}
  input{width:200px;border-radius:2px;margin:5px;padding: 3px;}
</style></head>
<body><p><input type="button" value="以垂直方式从右到左显示" id="myBtnVerticalRL"/><input type=
"button" value="以垂直方式从左到右显示" id="myBtnVerticalLR"/></p>
<div><h4>凉州词</h4><p>黄河远上白云间,<br>一片孤城万仞山。<br>羌笛何须怨杨柳,<br>春风不度玉
门关。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，writing-mode 用于控制文本的显示方向，该属性原本是控制内联元素的显示的（即所谓的文本布局 Text Layout），因为在亚洲，尤其像中国这样的东亚国家，有时文字的排版不是水平式的，而是垂直式的，例如中国的古诗、古文。writing-mode 属性主要有下列 3 个属性值。

（1）horizontal-tb：该属性值是默认值，表示文本流是水平方向（horizontal）的，内容是从上往下（tb：top-bottom）堆叠的。

（2）vertical-rl：该属性值表示文本流以垂直方向（vertical）展示，阅读的顺序是从右往左（rl：right-left），与中国古诗的阅读顺序一致。

（3）vertical-lr：该属性值表示文本流以垂直方向（vertical）展示，阅读的顺序是默认的从左往右（lr：left-right），也就是仅仅水平变垂直。

此实例的源文件名是 myHtmlB134.html。



图 053-1



图 053-2

054 绘制不规则图形实现不规则文字布局

此实例主要通过设置元素的 shape-outside 和 clip-path 属性以不规则的图形实现不规则的文字布局。当在 Google Chrome 浏览器中显示该页面时，以不规则的图形实现的文字布局效果如图 054-1 所示。有关此实例的主要代码如下。



图 054-1

```
<!doctype html><html><head><meta charset="UTF-8">
<style>
  * { margin: 0px; padding: 0px; position: relative;box-sizing: border-box;}
  .container { width: 100vw; height: 100vh;
               overflow: hidden;background-color: aqua;}
  .shaped {background: url("img/B114.jpg") no-repeat top/cover black;
           width: 60vw;height: 100vh;float: right; -webkit-shape-margin: 20px;
           -webkit-shape-outside: polygon(0 0, 100% 0, 100% 100%, 90% 100%);
           -webkit-clip-path: polygon(0 0, 100% 0, 100% 100%, 80% 100%);}
  .content {padding: 10px;text-align: justify;color: black; font-size: 15px;}
  .content h1 {width: 120px;height: 300vh;line-height: 1; float: left;
               padding-top: 100px;font-size: 30px;margin-top: 0px;
               shape-outside: polygon(0 0, 100% 0, 50% 100%, 0 100%); }
  p {text-indent: 1px;}
</style></head>
<body><div class="container"><div class="shaped"></div>
 <div class="content"><h1>园博园</h1>
   <p>园博园位于重庆市北部新区龙景湖公园，四面临街，可远眺缙云山、鸡公山，嘉陵江温塘峡、观音峡等山
景、水景、峡景和北碚城市景观，可满足游览休息。园博园，是一个集自然景观和人文景观为一体的超大型城市
生态公园。2014 年 5 月，重庆园博园风景区正式被国家旅游局批准为国家 AAAA(4A)级风景名胜区。</p></div>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，shape-outside 属性值用于创建不规则图形。在一般情况下，shape-outside 属性只能被应用于浮动元素，并且只能应用于块级元素，如果需要在非块级元素上使用这些属性，必须先把元素声明为块级。shape-* 值有 3 种赋值方式，即自动、基本图形或者图片链接。如果被设置为自动，浮动元素将继续作为传统的盒子模型进行渲染。绘制方法包括 rectangle、inset-rectangle、circle、ellipse、polygon 等。如果属性被设置为图片链接，浏览器

会按照图片的 alpha 通道来绘制图形形状。

此实例的源文件名是 myHtmlB114.html。

055 在段落文本的右上角放置文字环绕图片

此实例主要使用 before 选择器预留空白空间放置图片，从而实现在段落文本的右上角放置文字环绕图片的效果。当在 Google Chrome 浏览器中显示该页面时，在段落文本的右上角放置文字环绕图片的效果如图 055-1 所示。有关此实例的主要代码如下。



图 055-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   * {margin: 0px;padding: 0px;position: relative;box-sizing: border-box;}
  .myBox {width: 410px;}
  .content {padding: 10px;text-align: justify; color: black; font-size: 15px;}
  .content::before {content: ""; display: block; float: right;
                   width: 240px;height: 195px;}
  p {text-indent: 1px; font-size: 13px;}          /*文本显示样式*/
  img {width: 236px; height: 191px;left: 170px;top: 10px;
       border-radius: 5px; position: absolute;}   /*图像显示样式*/
  body {background-color: aquamarine;}
</style></head>
<body><div class="myBox"><img src="img/B117.jpg"/>
 <div class="content"><p>黄山,位于安徽省黄山市,原名黟山,唐朝时更名为黄山,取自"黄帝之山"之意。
黄山是世界自然和文化双遗产,世界地质公园,中国十大名胜古迹之一,国家 5A 级旅游景区。黄山风景区面积
160.6 平方千米,东起黄狮,西至小岭脚,北始二龙桥,南达汤口镇,分为温泉、云谷、玉屏、北海、松谷、钓桥、浮溪、
洋湖、福固九个管理区,包括 200 多个大小景点。黄山以奇松、怪石、云海、温泉、冬雪"五绝"著称于世,拥有"天
下第一奇山"之称。"五岳归来不看山,黄山归来不看岳"是对黄山最好的评价。</p></div></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content::before 中的 width 和 height 属性值通常与 img 中的 width 和 height 属性值一致，因为它们表示相同的空间位置和大小；content::before 中的 float: right 表示预留空间在右边。

此实例的源文件名是 myHtmlB117.html。

056 在段落文本的左上角放置文字环绕图片

此实例主要使用 before 选择器预留空白空间放置图片，从而实现在段落文本的左上角放置文字环绕图片的效果。当在 Google Chrome 浏览器中显示该页面时，在段落文本的左上角放置文字环绕图片的效果如图 056-1 所示。有关此实例的主要代码如下。



图 056-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  * { margin: 0px; padding: 0px; position: relative;box-sizing: border-box;}
  /*图文混合放置块*/
  .myBox {width: 410px;}
  /*段落文本显示样式*/
  .content {padding: 10px;text-align: justify;color: black;font-size: 15px;}
  .content::before { content: ""; display: block; float: left;
                  width: 225px; height: 165px;}
  p { text-indent: 1px; font-size: 13px;}            /*文本显示样式*/
  img { width: 215px;height: 155px;left: 10px;top: 10px;
        border-radius: 5px; position: absolute; }    /*图像显示样式*/
  body { background-color: aquamarine;}
</style></head>
<body><div class="myBox"><img src="img/B118.jpg"/>
 <div class="content"><p>乐山大佛，地处四川省乐山市，岷江、青衣江和大渡河三江汇流处，与乐山城隔江相望，北距成都 160 余公里。它是依凌云山栖霞峰临江峭壁凿造的一尊大佛，始凿于唐开元元年(公元 713 年)，历时 90 余年方建成，建高 71 米，有"山是一尊佛，佛是一座山"之称，是世界上最大的石刻大佛。乐山大佛头与山齐，足踏大江，双手抚膝，大佛体态匀称，神势肃穆，依山凿成，临江危坐。大佛通高 71 米，头高 14.7 米，头宽 10 米，发髻 1051 个，耳长 6.7 米，鼻和眉长 5.6 米，嘴巴和眼长 3.3 米，颈高 3 米，肩宽 24 米，手指长 8.3 米，从膝盖到脚背 28 米，脚背宽 9 米，脚面可围坐百人以上。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content::before 中的 width 和 height 属性值表示预留的放置图片的空间，因此它通常与 img 中的 width 和 height 属性值近似；content::before 中的 float:left 表示预留空间在左边。注意，content::before 中的 content:""看起来没有内容，但不能删除。

此实例的源文件名是 myHtmlB118.html。

057 在段落文本的左下角放置文字环绕图片

此实例主要使用 before 选择器预留空白空间放置图片，从而实现在段落文本的左下角放置文字环绕椭圆图片的效果。当在 Google Chrome 浏览器中显示该页面时，在段落文本的左下角放置文字环绕椭圆图片的效果如图 057-1 所示。有关此实例的主要代码如下。



图 057-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   * {margin: 0px;padding: 0px;position: relative;box-sizing: border-box;}
  .myBox {width: 450px;}
  .content {padding: 10px;text-align: justify;color: black;font-size: 15px;}
  .content::before {content: "";display: block; float: left;
            width: 450px;height: 350px; background-size: 100% 100%;
            shape-outside: ellipse(150px 100px at 150px 180px);
            -webkit-clip-path: ellipse(140px 90px at 150px 180px);
            background-image: url(img/B119.jpg);background-repeat: no-repeat;}
  p {margin: 5px;text-indent: 1px;font-size: 13px;}
  body {background-color: aquamarine;}
</style></head>
<body><div class="myBox"><div class="content">
    <p>大雁塔位于唐长安城晋昌坊(今陕西省西安市南)的大慈恩寺内,又名"慈恩寺塔"。唐永徽三年(652年),玄奘为保存由天竺经丝绸之路带回长安的经卷佛像主持修建了大雁塔,最初五层,后加盖至九层,再后层数和高度又有数次变更,最后固定为今天所看到的七层塔身,通高64.517米,底层边长25.5米。大雁塔作为现存最早、规模最大的唐代四方楼阁式砖塔,是佛塔这种古印度佛寺的建筑形式随佛教传入中原地区,并融入华夏文化的典型物证,是凝聚了汉族劳动人民智慧结晶的标志性建筑。1961年3月4日,国务院公布大雁塔为第一批全国重点文物保护单位。2014年6月22日,在卡塔尔多哈召开的联合国教科文组织第38届世界遗产委员会会议上,大雁塔作为中国、哈萨克斯坦和吉尔吉斯斯坦三国联合申遗的"丝绸之路:长安-天山廊道的路网"中的一处遗址点成功列入«世界遗产名录»。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content::before 中的 width 和 height 属性值表示预留的空间；shape-outside：ellipse(150px 100px at 150px 180px)中的 150px 100px 分别表示空白椭圆的两个半径，150px 180px 表示空白椭圆的圆心坐标，该空白椭圆用于控制

文本沿着椭圆边缘进行换行；-webkit-clip-path：ellipse(140px 90px at 150px 180px)中的140px 90px分别表示裁剪椭圆的两个半径，150px 180px表示裁剪椭圆的圆心坐标，该裁剪椭圆即是实例中显示的椭圆图片。注意，content::before中的content："" 看起来没有内容，但不能删除，它起着占位的作用。

此实例的源文件名是myHtmlB119.html。

058　对所有段落的第一个字设置加大、下沉效果

此实例主要在first-letter选择器中把首字的font-size值设置较大，然后通过float实现对所有段落的第一个字以加大、下沉效果显示。当在Google Chrome浏览器中显示该页面时，3个段落的第一个字加大、下沉的效果如图058-1所示。有关此实例的主要代码如下。

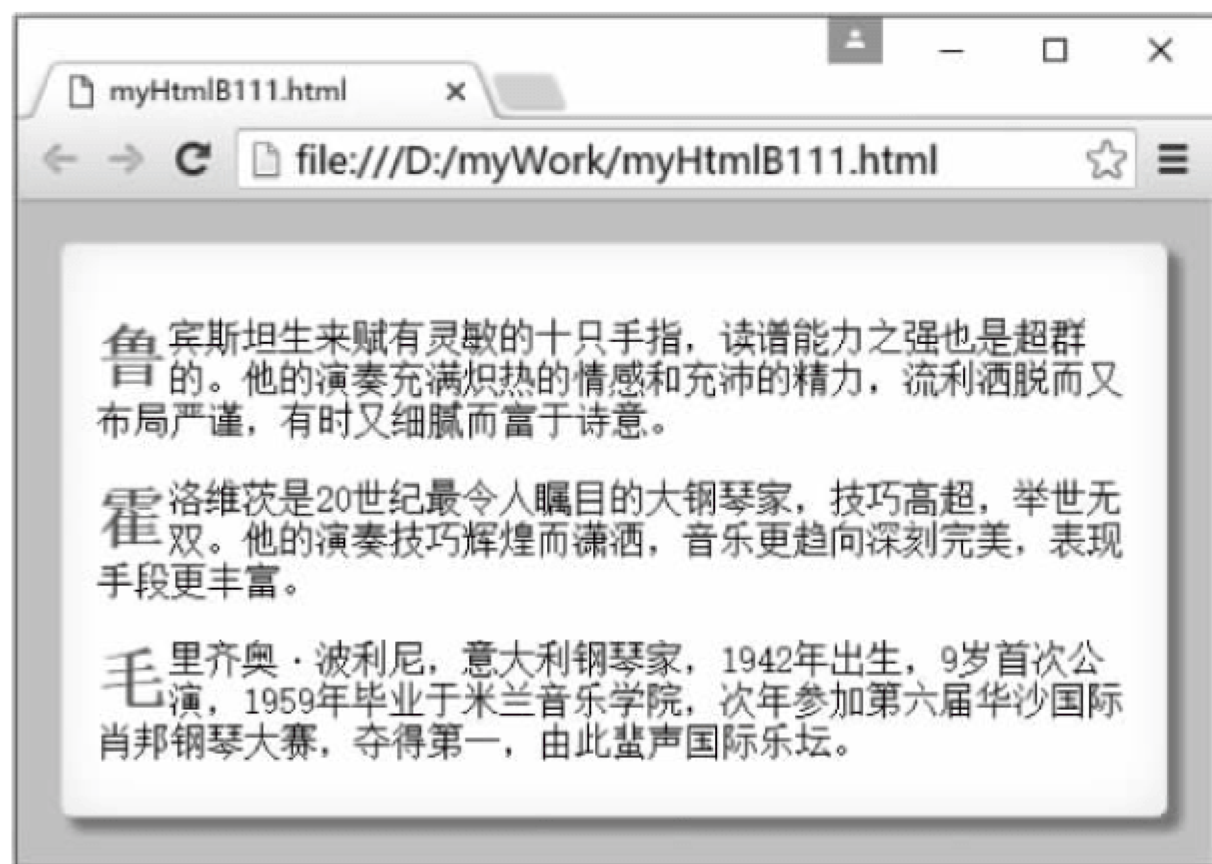


图　058-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block;width: 450px; height: 218px;padding: 15px;
       margin: 10px; background-color: #FFF; border: 1px solid #EEE;
       box-shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset;
       position: relative;border-radius: 5px;}
   p::first-letter { color: red;float: left;font-size: 30px;font-weight: bold;}
   body { background-color: lightgray;}
</style></head>
<body><div><p>鲁宾斯坦生来赋有灵敏的十只手指,读谱能力之强也是超群的。他的演奏充满炽热的情感和充沛的精力,流利洒脱而又布局严谨,有时又细腻而富于诗意。</p>
    <p>霍洛维茨是20世纪最令人瞩目的大钢琴家,技巧高超,举世无双。他的演奏技巧辉煌而潇洒,音乐更趋向深刻完美,表现手段更丰富。</p>
    <p>毛里齐奥·波利尼,意大利钢琴家,1942年出生,9岁首次公演,1959年毕业于米兰音乐学院,次年参加第六届华沙国际肖邦钢琴大赛,夺得第一,由此蜚声国际乐坛。</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，first-letter选择器用于选取指定其他选择器的首字母，可在first-letter选择器中使用的属性有字体属性、颜色属性、背景属性、外边距属性、内边距属性、边框属性、text-decoration、float、vertical-align、text-transform、line-height、clear等。

此实例的源文件名是myHtmlB111.html。

059 仅对第一个段落的首字设置加大、下沉效果

此实例主要通过定制 p:first-child::first-letter 选择器实现仅对第一个段落的首字设置加大、下沉效果。当在 Google Chrome 浏览器中显示该页面时，在 3 个段落中只有第一个段落的首字显示加大、下沉效果，如图 059-1 所示。有关此实例的主要代码如下。

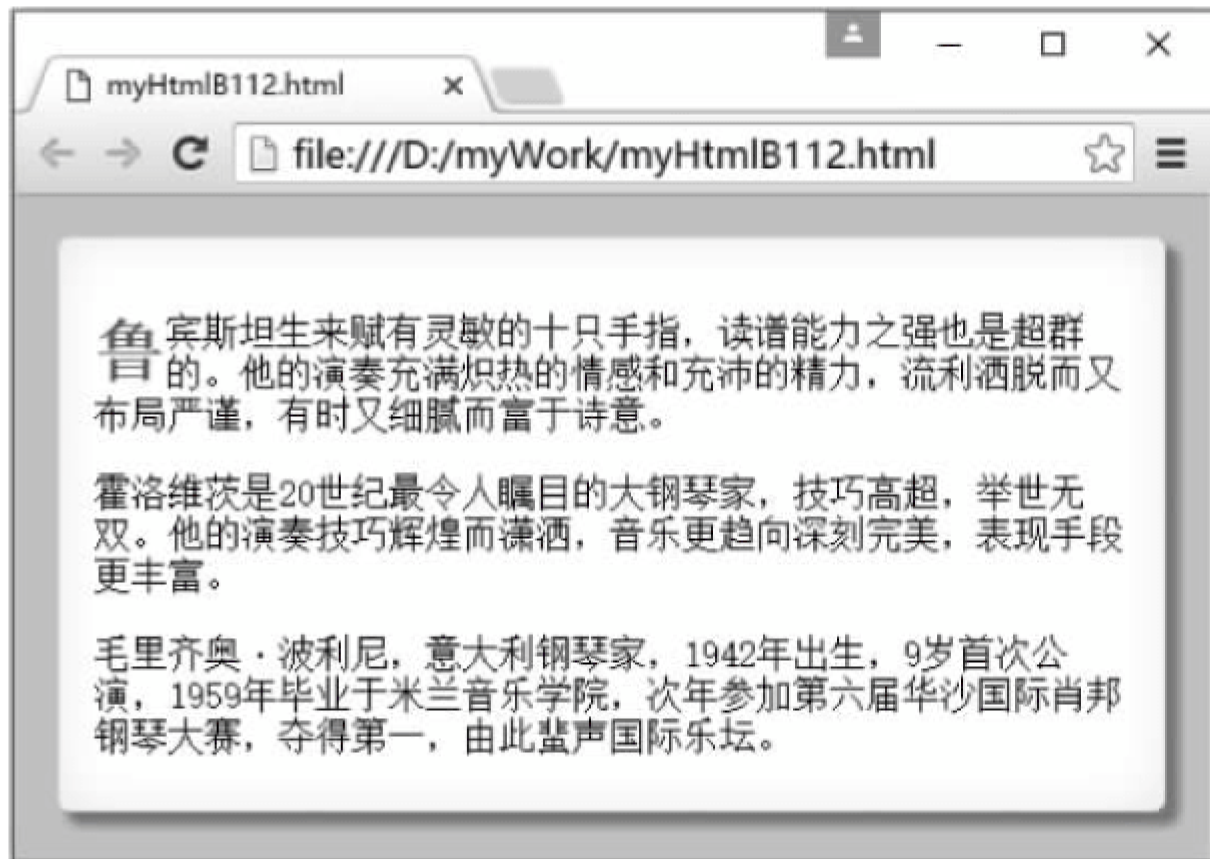


图 059-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block;width: 450px;height:218px;padding: 15px;
       margin:10px;background-color: #FFF;border: 1px solid #EEE;
       box-shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset;
       position: relative;border-radius: 5px;}
  p:first-child::first-letter {color: red; float: left;
                               font-size: 30px;font-weight: bold;}
  body { background-color: lightgray;}
</style></head>
<body><div><p>鲁宾斯坦生来赋有灵敏的十只手指,读谱能力之强也是超群的。他的演奏充满炽热的情感和充沛的精力,流利洒脱而又布局严谨,有时又细腻而富于诗意。</p>
    <p>霍洛维茨是 20 世纪最令人瞩目的大钢琴家,技巧高超,举世无双。他的演奏技巧辉煌而潇洒,音乐更趋向深刻完美,表现手段更丰富。</p>
    <p>毛里齐奥·波利尼,意大利钢琴家,1942 年出生,9 岁首次公演,1959 年毕业于米兰音乐学院,次年参加第六届华沙国际肖邦钢琴大赛,夺得第一,由此蜚声国际乐坛。</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，p:first-child::first-letter 选择器用于获取所有段落组成的集合中的第一个段落的首字(母)。

此实例的源文件名是 myHtmlB112.html。

060 在段落的第一个字的外围设置阴影效果

此实例主要通过 div:first-letter 选择器在 div 中查找第一个字，从而实现在段落的第一个字的外围设置阴影效果。当在 Google Chrome 浏览器中显示该页面时，两张简历的第一个字“徐”的外围出

现的阴影效果如图 060-1 所示。有关此实例的主要代码如下。



图 060-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block; width: 200px;height: 218px;
       margin: 10px; background-color: #FFF; border: 1px solid #EEE;
       box-shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset;
       position: relative;border-radius:5px;}
  div:first-letter{ font-size: 24px;background-color:white;
                    border-radius:5px;box-shadow: 0px 0px 10px 5px gray;}
  p{padding-left: 15px; padding-right: 10px; font-size: 14px;}
  body{background-color: lightgray;}
</style></head>
<body><div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县
滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中
国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div>
<div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会
夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红
军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,div:first-letter 选择器用于查找 div 的第一个字,box-shadow 用于向元素添加一个或多个阴影,box-shadow 的语法格式如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

在此实例中,box-shadow: 0px 0px 10px 5px gray 的两个 0px 分别代表 x-offset 和 y-offset,10px 表示 blur-radius,5px 表示 spread-radius,gray 表示 color。

此实例的源文件名是 myHtmlB052.html。

061 在段落的第一行字的外围设置阴影效果

此实例主要通过 div:first-line 选择器在 div 中查找第一行文本,从而实现在段落的第一行字的外围设置阴影效果。当在 Google Chrome 浏览器中显示该页面时,第一行诗的外围出现的阴影效果如图 061-1 所示。有关此实例的主要代码如下。



图 061-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block;width: 350px;height: 218px; margin: 10px;
       background-color: #FFF; border: 1px solid #EEE; border-radius: 5px;
       box-shadow: 5px 5px 5px 1px #999;position: relative;}
  div:first-line { font-size: 20px;background-color: white;
                   box-shadow: 0px 0px 10px 5px gray;}
  p {padding-left: 15px;padding-right: 10px;font-size: 20px;}
  body {background-color: lightgray;}
</style></head>
<body>
<div><p>汉皇重色思倾国,御宇多年求不得。<br> 杨家有女初长成,养在深闺人未识。<br>
          天生丽质难自弃,一朝选在君王侧。<br> 回眸一笑百媚生,六宫粉黛无颜色。<br>
          春寒赐浴华清池,温泉水滑洗凝脂。<br> 侍儿扶起娇无力,始是新承恩泽时。<br>
          云鬓花颜金步摇,芙蓉帐暖度春宵。<br> 春宵苦短日高起,从此君王不早朝。<br></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，div:first-line 选择器用于查找 div 的第一行文本，box-shadow 用于向元素添加一个或多个阴影。

此实例的源文件名是 myHtmlB053.html。

第2部分 图像

062 通过逐点处理像素实现图像底片效果

此实例主要对图像的每个像素取对立颜色值，从而实现对图像进行类似于底片的反色特效处理。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“对图像进行反色特效处理”按钮，将在下面显示经过反色特效处理之后的图像，如图 062-1 所示。有关此实例的主要代码如下。



图　062-1

```
<!doctype html><html><head><meta charset = UTF - 8 >
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function(){
   $("#myBtn").click(function() {       //对图像进行反色特效处理
    var myCanvas = $("#myCanvas").get(0);
```

```
    var myContext = myCanvas.getContext("2d");
    var myImage = $ ("#myJpg").get(0);
    myContext.drawImage(myImage,0,0);
    var myImgData = myContext.getImageData(0,0,myImage.width,myImage.height);
    var myLength = myImgData.data.length;
    for (var i = 0;i < myLength;i += 4){
     myImgData.data[i] = 255 - myImgData.data[i];
     myImgData.data[i + 1] = 255 - myImgData.data[i + 1];
     myImgData.data[i + 2] = 255 - myImgData.data[i + 2];
     myImgData.data[i + 3] = 255;
    }
    myContext.putImageData(myImgData,0,0); });});
</script></head>
<body><p>原始图像: </p><img id = "myJpg" src = "img/a030.jpg" />
<p><input type = "button" value = "对图像进行反色特效处理" id = "myBtn" style = "width:400px"></p>
<canvas id = "myCanvas" width = "900" height = "950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，getImageData()方法用于获取ImageData对象，该对象复制了画布指定矩形的像素数据ImageData。ImageData对象中的每个像素存储了4种信息，即RGBA值，其中R表示红色(0～255)，G表示绿色(0～255)，B表示蓝色(0～255)，A表示alpha通道(0～255，0是透明的，255是完全可见的)，color/alpha以数组形式存在，并存储于ImageData对象的data属性中。此实例中的for循环即是对color/alpha数组中的颜色值逐一取对立值。getImageData()方法的语法声明如下。

```
getImageData(x,y,width,height);
```

其中，参数x表示获取图像的左上角位置的x坐标；参数y表示获取图像的左上角位置的y坐标；参数width表示将要复制的矩形区域的宽度；参数height表示将要复制的矩形区域的高度。

putImageData()方法用于在画布上显示图像数据(从指定的ImageData对象)，该方法的语法声明如下。

```
putImageData(imgData,x,y,dirtyX,dirtyY,dirtyWidth,dirtyHeight);
```

其中，参数imgData规定要放回画布的ImageData对象；参数x表示ImageData对象左上角的x坐标，以像素计算；参数y表示ImageData对象左上角的y坐标，以像素计算；参数dirtyX是可选参数，以像素计算，表示在画布上放置图像的水平位置；参数dirtyY是可选参数，以像素计算，表示在画布上放置图像的垂直位置；参数dirtyWidth是可选参数，表示在画布上绘制图像所使用的宽度；参数dirtyHeight是可选参数，表示在画布上绘制图像所使用的高度。

此实例的源文件名是myHtmlA030.html。

063　采用平均值法将图像从彩色变为灰度

此实例主要通过对图像的每个像素的R、G、B取平均值实现将彩色图像转变为灰度图像。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“将彩色图像转变为灰度图像”按钮，将在下面显示经过处理之后的灰度图像，如图063-1所示。有关此实例的主要代码如下。



图　063-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {        //将彩色图像转变为灰度图像
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    var myImage=$("#myJpg").get(0);
    myContext.drawImage(myImage,0,0);
    var myImgData=myContext.getImageData(0,0,myImage.width,myImage.height);
    var myLength=myImgData.data.length;
    for (var i=0;i<myLength;i+=4){
     //取该像素点的 R、G、B 的平均值
     var myPixel=(myImgData.data[i]+myImgData.data[i+1]+myImgData.data[i+2])/3;
     //使用平均值重新设置该像素点的 R、G、B
     myImgData.data[i]=myImgData.data[i+1]=myImgData.data[i+2]=myPixel;
     //设置为不透明
     myImgData.data[i+3]=255;
    }
    myContext.putImageData(myImgData,0,0); });});
</script></head>
<body><p>原始图像:</p><img id="myJpg" src="img/a042.jpg" />
<p><input type="button" value="将彩色图像转变为灰度图像" id="myBtn" style="width:385px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，getImageData()方法用于获取ImageData对象，该对象复制了画布指定矩形的像素数据ImageData；putImageData()方法用于在画布上显示图像数据（从指定的ImageData对象）。将彩色图像转变成灰度图像通常有3种方法，即最大值法、平均值法和加权平均值法。这3种方法可以根据对图像灰度的具体要求进行选择，通常采用

平均值法。3 种方法的原理如下。

(1) 最大值法：最大值法使每个像素点的 R、G、B 值等于原像素点的 RGB 值中最大的一个，即 R=G=B=MAX(R,G,B)。最大值法会形成亮度很高的灰度图像。

(2) 平均值法：平均值法使每个像素点的 R、G、B 值等于原像素点的颜色值中的 RGB 值的平均值，即 R=G=B=(R+G+B)/3。平均值法会形成比较柔和的灰度图像。

(3) 加权平均值法：加权平均值法根据需要指定每个像素点 R、G、B 值的权数，并取其加权平均值，即 R=G=B=(Wr * R+Wg * G+Wb * B)/3。其中，Wr 代表该像素点 R 的权数，大于 0；Wg 代表该像素点 G 的权数，大于 0；Wb 代表该像素点 B 的权数，大于 0。通过对 Wr、Wg、Wb 取不同的值可以获得不同的效果。

在 ImageData 像素数据中，ImgData. data[i]、ImgData. data[i+1]、ImgData. data[i+2]分别代表每个像素点的 R、G、B 值，此实例的 var myPixel=(myImgData. data[i]+ myImgData. data[i+1]+myImgData. data[i+2])/3 表示取 i 像素点的 R、G、B 值的平均值，然后通过代码 myImgData. data[i]=myImgData. data[i+1]=myImgData. data[i+2]=myPixel 将 i 像素点的 R、G、B 值都设置为该平均值。ImgData. data[i+3]表示该像素点的透明度，所有像素点均设置为不透明(255)。

此实例的源文件名是 myHtmlA042. html。

064 使用拉普拉斯模板实现锐化处理图像

此实例主要通过使用拉普拉斯模板进行数学运算，从而实现对图像进行锐化特效处理。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“对图像进行锐化特效处理”按钮，将在下面显示经过锐化特效处理之后的图像，如图 064-1 所示。有关此实例的主要代码如下。



图 064-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function myEffect(myContext, myData) {                            //锐化特效
   var myTemp=myContext.createImageData(myData.width, myData.height);
   myTemp.data.set(myData.data);
   var myLaplacian=[ -1, -1, -1, -1, 9, -1, -1, -1, -1 ];  //拉普拉斯模板
   for ( var x=0; x<myTemp.width; x++){
    for ( var y=0; y<myTemp.height; y++) {
     var r=0, g=0, b=0;
     var Index=0;
     var a=0;
     for (var col=-1; col<=1; col++) {
      for (var row=-1; row<=1; row++) {                       //像素在数组中的索引
       var idx=((x+row)+(y+col)*myTemp.width)*4;
       r+=myTemp.data[idx+0]*myLaplacian[Index];
       g+=myTemp.data[idx+1]*myLaplacian[Index];
       b+=myTemp.data[idx+2]*myLaplacian[Index];
       Index++;
     } }
     r=r>255 ? 255 : r;                                       //处理颜色值溢出
     r=r<0 ? 0 : r;
     g=g>255 ? 255 : g;
     g=g<0 ? 0 : g;
     b=b>255 ? 255 : b;
     b=b<0 ? 0 : b;
     var midx=((x-1)+(y-1)*myTemp.width)*4;                  //在图像中设置新像素
     myData.data[midx+0]=r;                                   //R 通道
     myData.data[midx+1]=g ;                                  //G 通道
     myData.data[midx+2]=b ;                                  //B 通道
     myData.data[midx+3]=255;                                 //alpha 通道
   } } }
   $(document).ready(function () {
    $("#myBtn").click(function () {                          //对图像进行锐化特效处理
     var myImage=document.getElementById("myJpg");
     var myCanvas=document.getElementById("myCanvas");
     myCanvas.width=myImage.clientWidth;
     myCanvas.height=myImage.clientHeight;
     myContext=myCanvas.getContext("2d");
     myContext.drawImage(myImage, 0, 0, myCanvas.width, myCanvas.height);
     var myData=myContext.getImageData(0, 0, myCanvas.width, myCanvas.height);
     myEffect(myContext, myData);
     myContext.putImageData(myData, 0, 0); });});
</script></head>
<body><p>原始图像:</p><img id="myJpg" src="img/a041.jpg"/>
<p><input type="button" value="对图像进行锐化特效处理" id="myBtn" style="width:380px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myEffect(myContext, myData)方法用于实现图像的锐化特效。图像的锐化实际上就是要突出图像中有关形体的边缘。所谓形体的边缘就是图像像素点的颜色值发生显著变化的地方，在图像的平缓区，这种颜色值的变化比较平缓，而在图像的边缘区域，这种变化相当明显。也就是说，在平缓区，相邻两像素点的颜色值的差值较小，而在边缘区域，相邻两像素点的颜色值的变化陡得多，因此如果在边缘区域处理这个数值则可以使这种效果更加突出，而在非边缘区域则变得较暗，即图像的锐化。通常，对图像进行锐化处理使用拉普拉斯模板。

此实例的源文件名是 myHtmlA054.html。

065 对彩色图像进行灰白浮雕的特效处理

此实例主要对每个像素点的前、后像素点进行数学运算，从而实现对彩色图像进行类似于灰白浮雕的特效处理。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“对图像进行浮雕特效处理”按钮，将在下面显示经过灰白浮雕特效处理之后的图像，如图 065-1 所示。有关此实例的主要代码如下。



图 065-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function myEffect(myContext, myData) {                    //生成浮雕特效
   var myTemp = myContext.createImageData(myData.width, myData.height);
   myTemp.data.set(myData.data);
   for (var x = 1; x < myTemp.width - 1; x++) {
    for (var y = 1; y < myTemp.height - 1; y++) {           //获取像素在数组中的索引
     var idx = (x + y * myTemp.width) * 4;
     var bidx = ((x - 1) + y * myTemp.width) * 4;
     var aidx = ((x + 1) + y * myTemp.width) * 4;
     //计算新的 RGB 值
     var myRed = myTemp.data[aidx + 0] - myTemp.data[bidx + 0] + 128;
     var myGreen = myTemp.data[aidx + 1] - myTemp.data[bidx + 1] + 128;
     var myBlue = myTemp.data[aidx + 2] - myTemp.data[bidx + 2] + 128;
     myRed = (myRed < 0) ? 0 : ((myRed > 255) ? 255 : myRed);
     myGreen = (myGreen < 0) ? 0 : ((myGreen > 255) ? 255 : myGreen);
     myBlue = (myBlue < 0) ? 0 : ((myBlue > 255) ? 255 : myBlue);
     //根据新像素值填充图像
     myData.data[idx + 0] = myRed;                          //R 通道
     myData.data[idx + 1] = myGreen;                        //G 通道
     myData.data[idx + 2] = myBlue;                         //B 通道
     myData.data[idx + 3] = 255;                            //alpha 通道
  } } }
  $(document).ready(function() {
   $("#myBtn").click(function() {                          //对图像进行浮雕特效处理
```

```
        var myImage = document.getElementById("myJpg");
        var myCanvas = document.getElementById("myCanvas");
        myCanvas.width = myImage.clientWidth;
        myCanvas.height = myImage.clientHeight;
        myContext = myCanvas.getContext("2d");
        myContext.drawImage(myImage, 0, 0,myCanvas.width, myCanvas.height);
        var myData = myContext.getImageData(0, 0,myCanvas.width, myCanvas.height);
        myEffect(myContext, myData);
        myContext.putImageData(myData, 0, 0); });});
</script></head>
<body><p>原始图像:</p><img id = "myJpg" src = "img/a030.jpg"/>
<p><input type = "button" value = "对图像进行浮雕特效处理" id = "myBtn" style = "width:400px"></p>
<canvas id = "myCanvas" width = "900" height = "950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，createImageData()方法用于创建新的空白 ImageData 对象，新对象的默认像素值为 transparent black。ImageData 对象中的每个像素都存在 4 个方面的信息，即 RGBA 值，其中 R 表示红色(0～255)，G 表示绿色(0～255)，B 表示蓝色(0～255)，A 表示 alpha 通道(0～255，0 是透明的，255 是完全可见的)，因此 transparent black 表示(0,0,0,0)。color/alpha 以数组形式存在，并且既然数组包含了每个像素的 4 条信息，因此数组的大小是 ImageData 对象的 4 倍。包含 color/alpha 信息的数组存储于 ImageData 对象的 data 属性中。createImageData()方法的语法声明如下。

```
createImageData(width,height)
```

其中，参数 width 表示 ImageData 对象的宽度，以像素计算；参数 height 表示 ImageData 对象的高度，以像素计算。

此实例的源文件名是 myHtmlA051.html。

066　对彩色图像进行模糊化的特效处理

此实例主要使用 5×5 矩阵对每个像素点进行数学运算，从而使清晰的彩色图像变得模糊起来。当在浏览器中显示该页面时，将在上面显示正常的图像；单击"对图像进行模糊特效处理"按钮，将在下面显示经过模糊特效处理之后的图像，如图 066-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function myEffect(myContext, myData) {        //生成模糊特效
   var myTemp = myContext.createImageData(myData.width, myData.height);
   myTemp.data.set(myData.data);
   var myRed = 0.0, myGreen = 0.0, myBlue = 0.0;
   for ( var x = 0; x < myTemp.width; x++) {      //获取每个像素在数组中的索引
    for ( var y = 0; y < myTemp.height; y++) {
     var myIndex = (x + y * myTemp.width) * 4;
     for(var myColumn = - 2; myColumn <= 2; myColumn++) {
      var myColumnOff = myColumn + x;
      if(myColumnOff < 0 || myColumnOff >= myTemp.width) {
        myColumnOff = 0;
      }
      for(var myRow = - 2; myRow <= 2; myRow++) {
       var myRowOff = myRow + y;
       if(myRowOff < 0 || myRowOff >= myTemp.height) { myRowOff = 0; }
```

```
        var myNewIndex = (myColumnOff + myRowOff * myTemp.width) * 4;
        var r = myTemp.data[myNewIndex + 0];
        var g = myTemp.data[myNewIndex + 1];
        var b = myTemp.data[myNewIndex + 2];
        myRed += r; myGreen += g; myBlue += b; } }
      var nr = (myRed/25.0);                  //计算新的 RGB 值
      var ng = (myGreen/25.0); var nb = (myBlue/25.0);
      myRed = 0.0; myGreen = 0.0; myBlue = 0.0;
      //为新图像设置像素
      myData.data[myIndex + 0] = nr;          //R 通道
      myData.data[myIndex + 1] = ng;          //G 通道
      myData.data[myIndex + 2] = nb;          //B 通道
      myData.data[myIndex + 3] = 255;         //alpha 通道
     } } }
    $(document).ready(function() {
     $("#myBtn").click(function() {        //对图像进行模糊特效处理
      var myImage = document.getElementById("myJpg");
      var myCanvas = document.getElementById("myCanvas");
      myCanvas.width = myImage.clientWidth;
      myCanvas.height = myImage.clientHeight;
      myContext = myCanvas.getContext("2d");
      myContext.drawImage(myImage, 0, 0, myCanvas.width, myCanvas.height);
      var myData = myContext.getImageData(0, 0, myCanvas.width, myCanvas.height);
      myEffect(myContext, myData);
      myContext.putImageData(myData, 0, 0); });});
  </script></head>
 <body><p>原始图像:</p><img id = "myJpg" src = "img/a030.jpg"/>
 <p><input type = "button" value = "对图像进行模糊特效处理" id = "myBtn" style = "width:400px"></p>
 <canvas id = "myCanvas" width = "900" height = "950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myEffect(myContext, myData)方法用于生成图像的模糊效果，它主要通过 5×5 矩阵对每个像素点进行数学运算，从而实现模糊效果。

此实例的源文件名是 myHtmlA052.html。



图 066-1

067　使用随机数对图像进行油画特效处理

此实例主要使用随机数对图像进行油画特效处理。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“对图像进行油画特效处理”按钮，将在下面显示经过油画特效处理之后的图像，如图 067-1 所示。有关此实例的主要代码如下。



图　067-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function myEffect(myContext, myData) {                    //油画特效
   var myTemp = myContext.createImageData(myData.width, myData.height);
   myTemp.data.set(myData.data);
   var myModel = 8; var x = myTemp.width - myModel;
   while(x > 1){ var y = myTemp.height - myModel;
    while(y > 1) {
     var myPos = parseInt(Math.floor(Math.random() * ( 10 + 1) % myModel));
     var idx = (x + y * myTemp.width) * 4;                //像素在数组中的索引
     var midx = ((x + myPos) + (y + myPos) * myTemp.width) * 4;
     //在图像中设置新像素
     myData.data[midx + 0] = myTemp.data[idx + 0];        //R 通道
     myData.data[midx + 1] = myTemp.data[idx + 1];        //G 通道
     myData.data[midx + 2] = myTemp.data[idx + 2];        //B 通道
     myData.data[midx + 3] = 255;                         //alpha 通道
     y = y - 1;
    } x = x - 1; }}
  $(document).ready(function() {
   $("#myBtn").click(function() {                         //对图像进行油画特效处理
```

```
        var myImage = document.getElementById("myJpg");
        var myCanvas = document.getElementById("myCanvas");
        myCanvas.width = myImage.clientWidth;
        myCanvas.height = myImage.clientHeight;
        myContext = myCanvas.getContext("2d");
        myContext.drawImage(myImage, 0, 0, myCanvas.width, myCanvas.height);
        var myData = myContext.getImageData(0, 0, myCanvas.width, myCanvas.height);
        myEffect(myContext, myData);
        myContext.putImageData(myData, 0, 0); });});
  </script></head>
 <body><p>原始图像：</p><img id = "myJpg" src = "img/a041.jpg"/>
 <p><input type = "button" value = "对图像进行油画特效处理" id = "myBtn" style = "width:380px"></p>
 <canvas id = "myCanvas" width = "900" height = "950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，Math.random()用于获取 0 和 1 之间的伪随机数，Math.floor(Math.random() * (10+1)%myModel)实际上就是取 10 范围内的数字除以 myModel 之后的余数。myModel 值越大，油画效果越明显；myModel 值越小，油画效果越不明显。当对彩色图像中的某范围内的像素进行随机数处理后，图像就会产生毛玻璃带水雾状的油画效果。

此实例的源文件名是 myHtmlA055.html。

068　使用随机数对图像进行雾化特效处理

此实例主要使用随机数对图像进行雾化特效处理。当在浏览器中显示该页面时，将在上面显示正常的图像；单击"对图像进行雾化特效处理"按钮，将在下面显示经过雾化特效处理之后的图像，如图 068-1 所示。有关此实例的主要代码如下。



图　068-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function myEffect(myContext, myData) {                    //雾化特效
   var myTemp = myContext.createImageData(myData.width, myData.height);
   myTemp.data.set(myData.data);
   for ( var x = 1; x < myTemp.width - 1; x++){
    for ( var y = 1; y < myTemp.height - 1; y++){
     var myPara = parseInt(Math.floor(Math.random() * ( 100 + 1)));
     var dx = x + myPara % 29; var dy = y + myPara % 29;
     if(dx >= myTemp.width)
      dx = myTemp.width - 1;
     if(dy >= myTemp.height)
      dy = myTemp.height - 1;
     var idx = (dx + dy * myTemp.width) * 4;
     var midx = (x + y * myTemp.width) * 4;
     //在图像中设置新像素
     myData.data[midx + 0] = myTemp.data[idx + 0];   //R 通道
     myData.data[midx + 1] = myTemp.data[idx + 1];   //G 通道
     myData.data[midx + 2] = myTemp.data[idx + 2];   //B 通道
     myData.data[midx + 3] = 255;                    //alpha 通道
  } } }
  $(document).ready(function() {
   $("#myBtn").click(function() {                    //对图像进行雾化特效处理
    var myImage = document.getElementById("myJpg");
    var myCanvas = document.getElementById("myCanvas");
    myCanvas.width = myImage.clientWidth;
    myCanvas.height = myImage.clientHeight;
    myContext = myCanvas.getContext("2d");
    myContext.drawImage(myImage, 0, 0,myCanvas.width, myCanvas.height);
    var myData = myContext.getImageData(0, 0,myCanvas.width, myCanvas.height);
    myEffect(myContext, myData);
    myContext.putImageData(myData, 0, 0);
  }); });
 </script></head>
<body><p>原始图像: </p><img id="myJpg" src="img/a041.jpg"/>
<p><input type="button" value="对图像进行雾化特效处理" id="myBtn" style="width:380px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，Math.random()用于获取 0 和 1 之间的伪随机数，Math.floor(Math.random() * (100＋1))实际上就是取 100 范围内的随机数。影响图像雾化效果的一个重要因素是像素块的大小，即 dx、dy 的波动范围，所选取的像素块越大，产生的雾化效果越明显；所选取的像素块越小，产生的雾化效果越不明显。

此实例的源文件名是 myHtmlA056.html。

069　选择不同的组合模式叠加显示两幅图像

此实例主要通过设置 globalCompositeOperation 属性实现以不同的组合模式叠加显示两幅图像。当在浏览器中显示该页面时，单击“显示第一幅图”按钮，则在下面显示第一幅图；单击“显示第二幅图”按钮，则在下面显示第二幅图；单击“显示叠加的两幅图”按钮，则第一幅图和第二幅图叠加之后的效果如图 069-1 所示。有关此实例的主要代码如下。



图 069-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnA").click(function() {          //显示第一幅图
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    myContext.globalCompositeOperation="source-over";
    var myImageA=new Image();
    myImageA.src="img/a048.jpg";
    myImageA.onload=function() {
     myContext.drawImage(myImageA, 0, 0, 350, 210);
   }; });
   $("#myBtnB").click(function() {          //显示第二幅图
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    myContext.globalCompositeOperation="source-over";
    var myImageB=new Image();
    myImageB.src="img/a041.jpg";
myImageB.onload=function() {
  myContext.drawImage(myImageB, 0, 0, 350, 210);
   }; });
   $("#myBtn").click(function() {     //显示叠加的两幅图
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    //显示第一幅图(源图像)
    var myImageA=new Image();
    myImageA.src="img/a048.jpg";
    myImageA.onload=function() { myContext.drawImage(myImageA,0,0,350, 210);};
    var myOptions=new Array("source-atop","source-in",
                           "source-out","source-over","destination-atop",
                           "destination-in", "destination-out",
                           "destination-over","lighter","copy","xor");
    i=8;
    //设置组合方式,选择 myOptions 数组的第 9 个组合方式 lighter
    myContext.globalCompositeOperation=myOptions[i];
    //显示第二幅图(目标图像)
    var myImageB=new Image();
    myImageB.src="img/a041.jpg";
```

```
    myImageB.onload = function() {
     myContext.drawImage(myImageB, 0, 0, 350, 210); };});});
</script></head>
<body>
<p><input type = "button" value = "显示第一幅图" id = "myBtnA" style = "width:105px">
   <input type = "button" value = "显示第二幅图" id = "myBtnB" style = "width:105px">
<input type = "button" value = "显示叠加的两幅图" id = "myBtn" style = "width:125px"></p>
<canvas id = "myCanvas" width = "400" height = "450"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，globalCompositeOperation 属性用于设置或返回如何将一个源（新的）图像绘制到目标（已有）图像上，该属性支持的选项及意义如表 069-1 所示。

表 069-1　globalCompositeOperation 属性支持的选项及意义

值	意　义
source-over	默认值，在目标图像上显示源图像
source-atop	在目标图像顶部显示源图像，源图像位于目标图像之外的部分是不可见的
source-in	在目标图像中显示源图像，只有目标图像内的源图像部分会显示，目标图像是透明的
source-out	在目标图像之外显示源图像，只会显示目标图像之外的源图像部分，目标图像是透明的
destination-over	在源图像上方显示目标图像
destination-atop	在源图像顶部显示目标图像，源图像之外的目标图像部分不会被显示
destination-in	在源图像中显示目标图像，只有源图像内的目标图像部分会被显示，源图像是透明的
destination-out	在源图像外显示目标图像，只有源图像外的目标图像部分会被显示，源图像是透明的
lighter	显示源图像＋目标图像
copy	显示源图像，忽略目标图像
xor	使用异或操作对源图像与目标图像进行组合

此实例的源文件名是 myHtmlA049.html。

070　选择不同的混合模式叠加显示两幅图像

此实例主要通过设置 globalCompositeOperation 属性实现以不同的混合模式叠加显示两幅图像。当在浏览器中显示该页面时，单击“显示第一幅图”按钮，则在下面显示第一幅图；单击“显示第二幅图”按钮，则在下面显示第二幅图；单击“显示混合的两幅图”按钮，则第一幅图和第二幅图混合之后的效果如图 070-1 所示。有关此实例的主要代码如下。



图　070-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnA").click(function() {      //显示第一幅图
     var myCanvas=$("#myCanvas").get(0);
     var myContext=myCanvas.getContext("2d");
     myContext.globalCompositeOperation="normal";
     var myImageA=new Image();
     myImageA.src="img/a050A.jpg";
     myImageA.onload=function() {
      myContext.drawImage(myImageA, 0, 0, 350, 210);
    };});
    $("#myBtnB").click(function() {      //显示第二幅图
     var myCanvas=$("#myCanvas").get(0);
     var myContext=myCanvas.getContext("2d");
     myContext.globalCompositeOperation="normal";
     var myImageB=new Image();
     myImageB.src="img/a050B.jpg";
     myImageB.onload=function() {
      myContext.drawImage(myImageB, 0, 0, 350, 210); }; });
    $("#myBtn").click(function() {       //显示混合的两幅图
     var myCanvas=$("#myCanvas").get(0);
     var myContext=myCanvas.getContext("2d");
     var myOptions=new Array("normal","darken","lighten",
        "multiply","screen","color-burn","color-dodge",
        "hard-light","soft-light","overlay","difference","exclusion");
     var myImageA=new Image();
     myImageA.src="img/a050A.jpg";
     myImageA.onload=function(){
      myContext.drawImage(myImageA,0, 0, 350, 210);
      myContext.globalCompositeOperation=myOptions[1];
      var myImageB=new Image();
      myImageB.src="img/a050B.jpg";
      myImageB.onload=function(){
       myContext.drawImage(myImageB,0, 30); };};});});
</script></head>
<body><p>
<input type="button" value="显示第一幅图" id="myBtnA" style="width:105px">
<input type="button" value="显示第二幅图" id="myBtnB" style="width:105px">
<input type="button" value="显示混合的两幅图" id="myBtn" style="width:125px"></p>
<canvas id="myCanvas" width="400" height="450"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，globalCompositeOperation 属性用于设置或返回如何将一个源(新的)图像绘制到目标(已有)图像上，该属性支持混合模式的选项，这些选项的意义如下。

(1) normal：默认值，该选项实现正常模式，不混合。

(2) darken：该选项实现变暗模式，逐像素对比基色和混合色，只保留较深的颜色，去除较浅的颜色(白色为最浅的颜色，黑色为最深的颜色)。

(3) lighten：该选项实现变亮模式，逐像素对比基色和混合色，只保留较浅的颜色，去除较深的颜色(白色为最浅的颜色，黑色为最深的颜色)。

(4) multiply：该选项实现正片叠底模式，逐像素对比基色与混合色，将基色的灰度级与混合色的

灰度级进行乘法计算，获得灰度级更低的颜色而成为合成后的颜色。任何颜色与黑色相乘得到黑色，任何颜色与白色相乘则保持颜色不变。与变暗混合模式不同，这种混合模式在变暗图像时图像暗部区域的过渡很平缓，有利于保持原始图像的轮廓与图像中的阴影部分。

（5）screen：该选项实现滤色模式，与正片叠底模式相反，滤色模式将上、下两层图层的像素颜色的灰度级进行乘法计算，获得灰度级更高的颜色而成为合成后的颜色。图层合成后的效果简单地说是高灰阶的像素显现而低灰阶不显现（即浅色出现，深色不出现），产生的图像更加明亮。

（6）color-burn：该选项实现颜色加深模式，在使用这种模式时会使图像颜色变得更暗，混合色越暗，效果越细腻。让图像颜色更暗有点类似于正片叠底，但不同的是它会根据混合色相应增加图像的对比度。注意，与白色混合没有任何效果。

（7）color-dodge：该选项实现颜色减淡模式，在使用这种模式时会使图像颜色变得更亮，混合色越亮，效果越细腻。这种模式与颜色加深模式正好相反，通过降低对比度、加亮底层颜色来反映混合色彩，和黑色混合没有任何效果。

（8）hard-light：该选项实现强光模式。在对两幅图像进行逐像素比较时，如果混合色的灰度级小于等于0.5，则采用正片叠底模式，否则采用滤色模式（不同像素所采用的模式可能不同），产生的效果就好像为图像应用强烈的聚光灯。如果用纯黑或者纯白进行混合，得到的也是纯黑或者纯白。

（9）soft-light：该选项将混合色以柔光的方式混合到基色中，当基色的灰阶趋于高或低时会调整颜色合成结果的阶调趋于中间的灰阶调，从而获得色彩较为柔和的合成效果，形成的结果是图像的中亮色调变得更亮，暗色区域变得更暗，图像反差增大，类似于柔光灯照射图像的效果。至于变暗还是提亮画面颜色，取决于混合色信息，如果混合色的灰度级大于0.5，基色会变亮；如果混合色的灰度级小于0.5，基色会变暗。

（10）overlay：该选项实现叠加模式。在对两幅图像进行逐像素比较时，如果基色的灰度级小于等于0.5，则采用正片叠底模式，否则采用滤色模式（不同像素所采用的模式可能不同）。一般来说，进行叠加模式的图像混合后中间色调均会产生变化，但高亮色和暗色基本保持不变，因此基色的高光与阴影部分的亮度细节就会被保留。其效果与强光模式正好相反。

（11）difference：该选项实现差值模式，将混合色与基色的RGB值中的每个值分别进行比较，用高值减去低值作用合成后的颜色，所以这种模式经常使用，白色与任何颜色混合得到反相色，黑色与任何颜色混合后颜色不变。

（12）exclusion：该选项实现排除模式，与差值模式的作用类似，只是排除模式的结果色的对比度没有差值模式强。白色与基色混合得到基色补色，黑色与基色混合得到基色。

此实例的源文件名是myHtmlA050.html。

071　在图像中抠取某部分并对其进行局部放大

此实例主要通过使用drawImage()方法实现抠取部分图像并对其进行局部放大显示。当在浏览器中显示该页面时，单击“抠取部分图像并进行放大显示”按钮，将在下面的左边显示原图，在右边显示原图中经过放大的头像，如图071-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $ (document).ready(function() {
   $ ("#myBtn").click(function() {                //抠取部分图像并进行放大显示
    var myCanvas = $ ("#myCanvas").get(0);
```

```
        var myContext = myCanvas.getContext("2d");
        var myImage = new Image();
        myImage.src = "img/a043.jpg";
        myImage.onload = function() {
         myContext.drawImage(myImage,0,0,200,200);     //显示原图
         myContext.drawImage(myImage,60,10,100,100,210,0,200,200);     //显示局部放大图
    } });});
</script></head>
<body><p><input type = "button" value = "抠取部分图像并进行放大显示" id = "myBtn" style = "width:
400px"></p>
<canvas id = "myCanvas" width = "400" height = "200" ></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，drawImage()方法用于在画布上绘制图像或视频，该实例主要通过在此方法中设置相关参数实现抠取图像的部分区域并放大显示。drawImage()方法的语法声明如下。

```
drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
```

其中，参数 img 规定要使用的图像、画布或视频；参数 sx 是可选参数，表示开始剪切的 x 坐标位置；参数 sy 是可选参数，表示开始剪切的 y 坐标位置；参数 swidth 是可选参数，表示被剪切图像的宽度；参数 sheight 是可选参数，表示被剪切图像的高度；参数 x 表示在画布上放置图像的 x 坐标位置；参数 y 表示在画布上放置图像的 y 坐标位置；参数 width 是可选参数，表示要使用的图像的宽度(伸展或缩小图像)；参数 height 是可选参数，表示要使用的图像的高度(伸展或缩小图像)。

此实例的源文件名是 myHtmlA044.html。

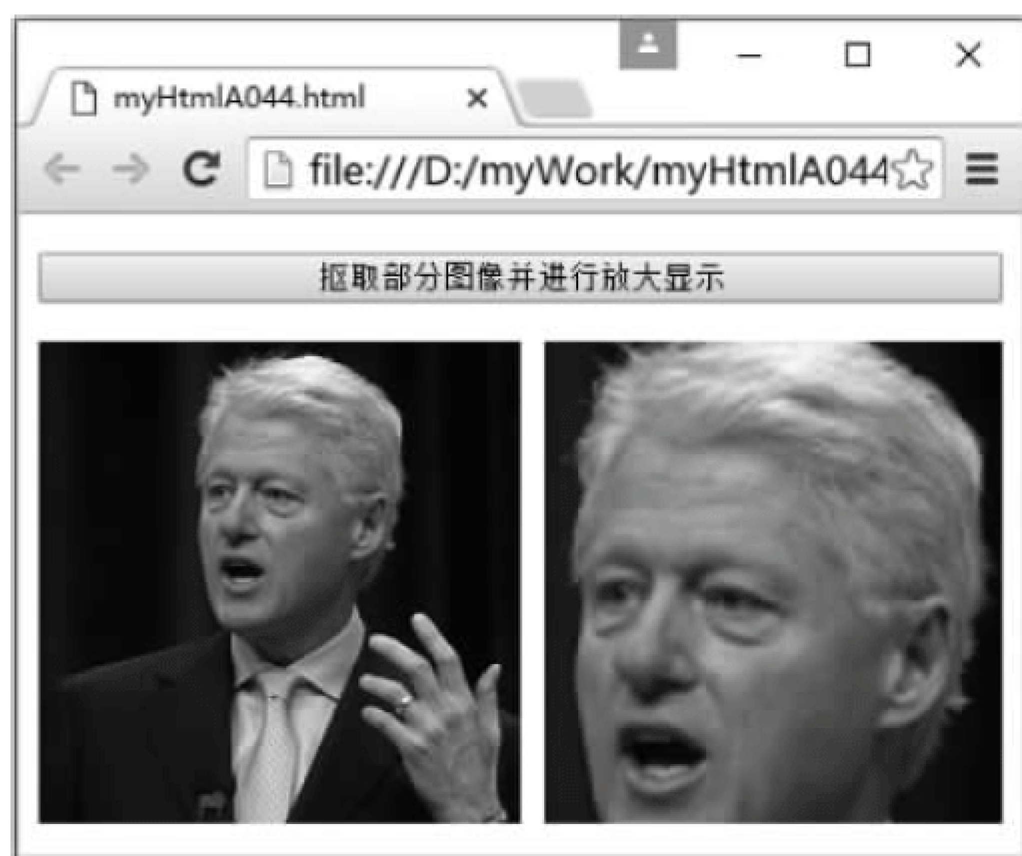


图 071-1

072 通过绘制五角星的形状来裁剪图像

此实例主要使用 lineTo()方法绘制五角星，然后使用 clip()方法按照五角星的形状裁剪图像。当在浏览器中显示该页面时，单击“按照五角星的形状裁剪图像”按钮，则在下面显示的图像将被裁剪成一个五角星，如图 072-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {          //按照五角星的形状裁剪图像
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    var myImage=new Image();
    myImage.src="img/a043.jpg";
    myImage.onload=function(){
     var n=0; var dx=100; var dy=0; var s=150;
     myContext.beginPath();
     myContext.translate(80,120);
     var x=Math.sin(0); var y=Math.cos(0);
     var myDig=Math.PI/5*4;
     for(var i=0; i<6; i++){          //创建五角星
      var x=Math.sin(i*myDig);
      var y=Math.cos(i*myDig);
      myContext.lineTo( dx+x*s,dy+y*s);
     }
     myContext.stroke();
     myContext.clip();                //根据五角星裁剪
     myContext.drawImage(myImage,-50,-150,300,300); };});});
 </script></head>
<body><p><input type="button" value="按照五角星的形状裁剪图像" id="myBtn" style="width:350px">
</p>
<canvas id="myCanvas" width="1900" height="1950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，lineTo()方法用于添加一个新点，然后创建从该点到画布中最后指定点的线条（该方法并不会实际绘制线条）。在该实例中如果删除myContext.drawImage(myImage,－50,－150,300,300)，即可看到绘制的线条。lineTo()方法的语法声明如下。

```
lineTo(x,y);
```

其中，参数 x 表示路径的目标位置的 x 坐标；参数 y 表示路径的目标位置的 y 坐标。

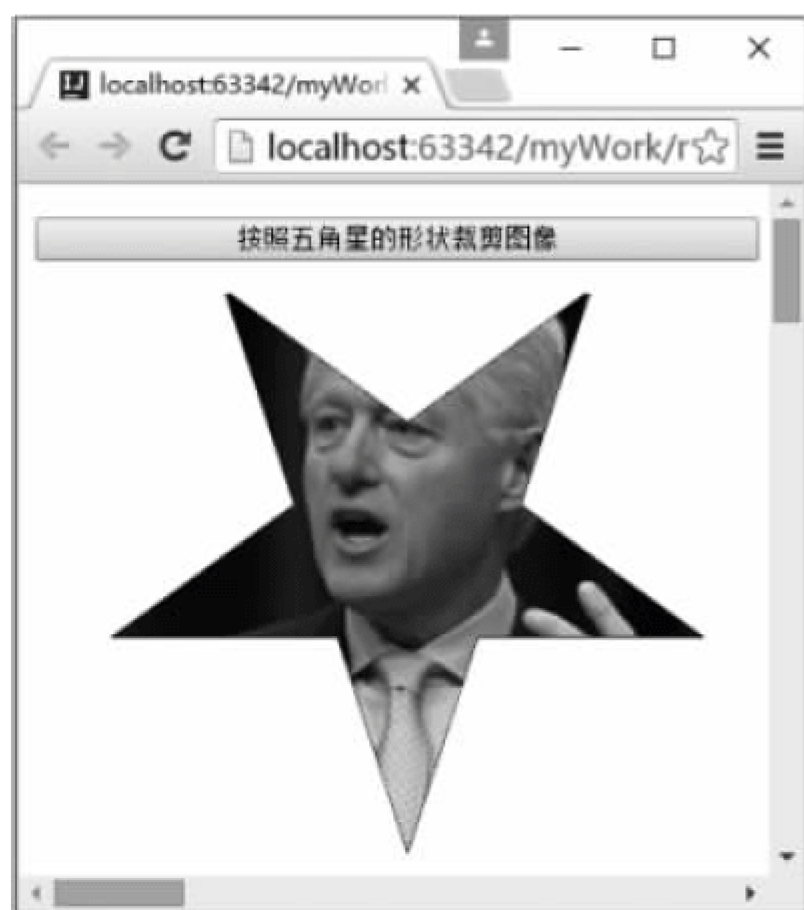


图　072-1

clip()方法用于从原始画布中剪切任意形状和尺寸，一旦剪切了某个区域，所有之后的绘图都会被限制在被剪切的区域内(不能访问画布上的其他区域)。

此实例的源文件名是 myHtmlA046.html。

073 通过绘制圆饼图的形状来裁剪图像

此实例主要使用 arc()方法绘制残缺的圆饼图，然后使用 clip()方法按照圆饼图的弧线形状裁剪图像。当在浏览器中显示该页面时，单击“按照弧线的形状裁剪图像”按钮，则在下面显示的图像将被裁剪成一个由弧线围成的封闭图形，如图 073-1 所示。有关此实例的主要代码如下。

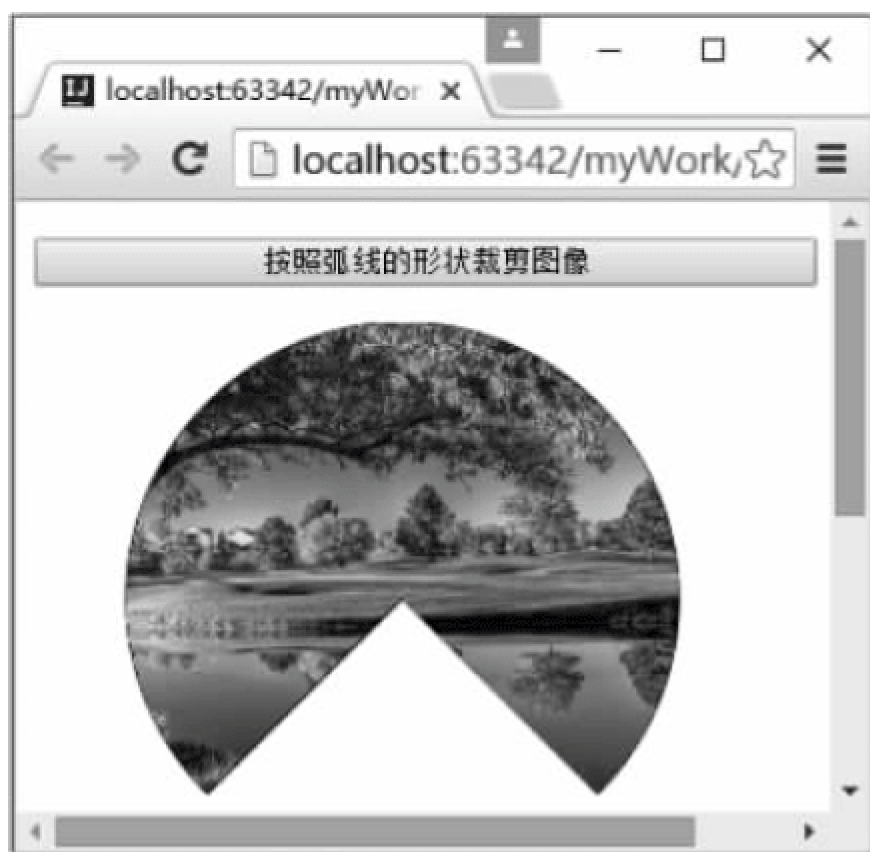


图 073-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {      //按照弧线的形状裁剪图像
     var myCanvas = $("#myCanvas").get(0);
     var myContext = myCanvas.getContext("2d");
     var myImage = new Image();
     myImage.src = "img/a048.jpg";
     myImage.onload = function(){
       myContext.beginPath();
       //绘制起始角为 135°、终止角为 45°的弧线,顺时针旋转
       myContext.arc(160,120,120,2 * Math.PI * 135/360,2 * Math.PI * 45/360);
       myContext.lineTo(160,120);
       myContext.closePath();
       myContext.stroke();
       myContext.clip();                //按照弧线的形状裁剪图像
       myContext.drawImage(myImage,0,0,350,210); };});});
 </script></head>
<body><p><input type = "button" value = "按照弧线的形状裁剪图像" id = "myBtn" style =
"width:340px"></p>
<canvas id = "myCanvas" width = "400" height = "450"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，arc()方法用于创建弧/曲线，或者用于创建圆/部分圆。如果需要使用 arc()方法创建圆，应把起始角设置为 0、结束角设置为 2 * Math.

PI。arc()方法的语法声明如下。

```
arc(x,y,r,sAngle,eAngle,counterclockwise);
```

其中,参数 x 表示圆心的 x 坐标;参数 y 表示圆心的 y 坐标;参数 r 表示圆的半径;参数 sAngle 表示起始角,以弧度计,弧(圆形)的三点(时)钟位置是 0°;参数 eAngle 表示结束角,以弧度计;参数 counterclockwise 是可选参数,规定应该逆时针还是顺时针绘图,false 表示顺时针,true 表示逆时针。

此实例的源文件名是 myHtmlA048.html。

074　采用均匀压缩法创建椭圆并裁剪图像

此实例主要通过使用 scale()方法等实现以均匀压缩的方式创建椭圆并裁剪图像。当在浏览器中显示该页面时,单击“按照椭圆的形状裁剪图像”按钮,则在下面显示的图像将被裁剪成一个椭圆,如图 074-1 所示。有关此实例的主要代码如下。



图　074-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {         //按照椭圆的形状裁剪图像
    var myCanvas = $("#myCanvas").get(0);
    var myContext = myCanvas.getContext("2d");
    var myImage = new Image();
    myImage.src = "img/a047.jpg";
    myImage.onload = function(){               //采用均匀压缩法绘制椭圆
      EvenCompEllipse(myContext, 174,102, 170, 100)
      myContext.clip();
      myContext.drawImage(myImage,0,0,350,210); };});
  function EvenCompEllipse(myContext, x, y, a, b){
    myContext.save();
    var r = (a > b) ? a : b;                   //选择 a、b 中的较大者作为 arc()方法的半径参数
    var ratioX = a/r;                          //横轴缩放比例
    var ratioY = b/r;                          //纵轴缩放比例
    myContext.scale(ratioX, ratioY);           //进行缩放(均匀压缩)
    myContext.beginPath();
```

```
        //从椭圆的左端点开始逆时针绘制
        myContext.moveTo((x + a)/ratioX, y/ratioY);
        myContext.arc(x/ratioX, y/ratioY, r, 0, 2 * Math.PI);
        myContext.closePath();
        myContext.stroke();
        myContext.restore(); };});
  </script></head>
<body><p><input type = "button" value = "按照椭圆的形状裁剪图像" id = "myBtn" style = "width:340px">
</p>
<canvas id = "myCanvas" width = "400" height = "450"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，EvenCompEllipse()自定义方法是用arc()方法绘制圆，结合 scale()方法在横轴或纵轴方向均匀压缩，从而产生椭圆的效果。scale()方法的作用是缩放当前绘图，使其更大或更小。如果对绘图进行缩放，所有之后的绘图也会被缩放，定位也会被缩放。例如，对于 scale(2,2)，绘图将定位于距离画布左上角两倍远的位置。scale()方法的语法声明如下。

```
scale(scalewidth,scaleheight);
```

其中，参数 scalewidth 表示缩放当前绘图的宽度(1=100%，0.5=50%，2=200%，以此类推)；参数 scaleheight 表示缩放当前绘图的高度(1=100%，0.5=50%，2=200%，以此类推)。

此实例的源文件名是 myHtmlA047.html。

075 通过绘制多个圆形实现太极图案的绘制

此实例主要使用 arc()方法绘制多个圆形，从而实现太极图案的绘制。当在浏览器中显示该页面时，单击“绘制太极图案”按钮，将在下面显示一个黑白的太极图案，如图 075-1 所示。有关此实例的主要代码如下。

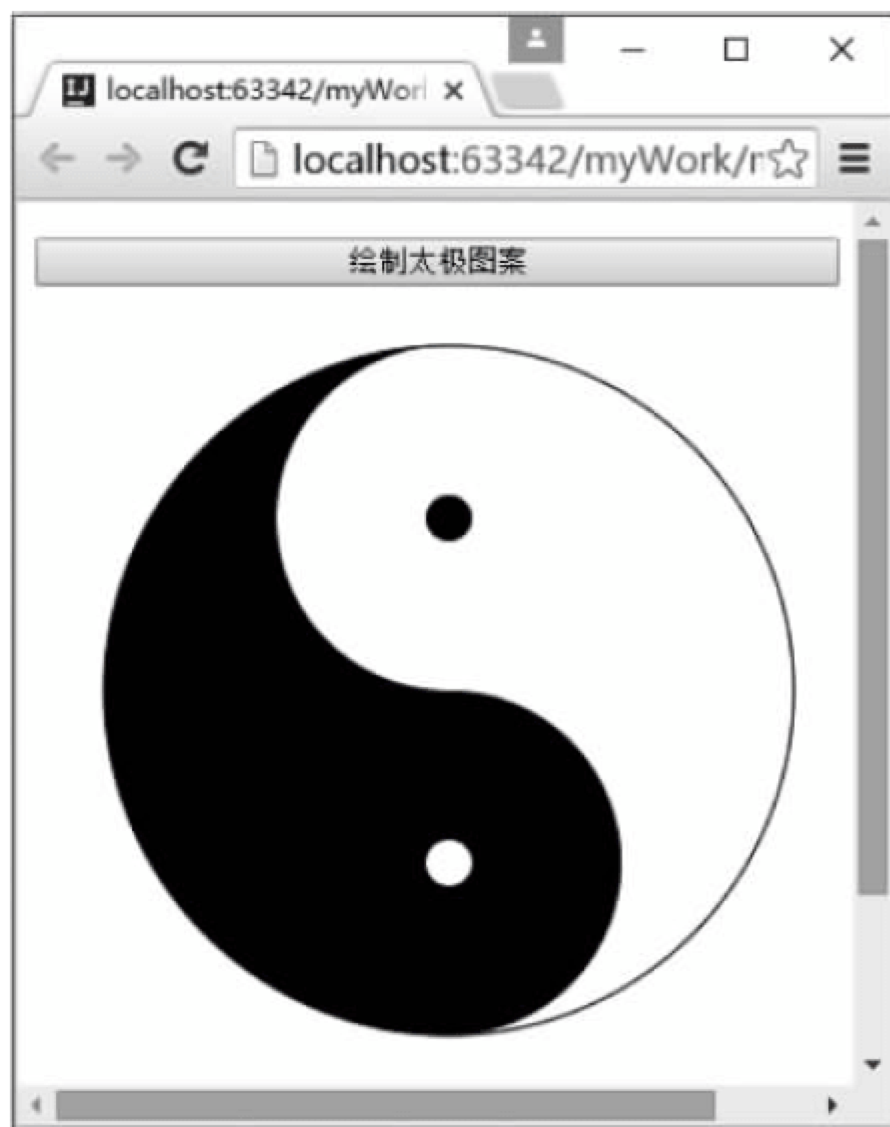


图 075-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {        //绘制太极图案
    var myCanvas=document.getElementById("myCanvas");
    myContext=myCanvas.getContext("2d");
    //重新映射画布上的(0,0)位置
    myContext.translate(-20,-40);
    //在右侧绘制白色半圆,可以省略
    myContext.fillStyle="#FFF";
    myContext.beginPath();
    myContext.arc(200, 200, 150, 1.5*Math.PI, Math.PI/2, false);
    myContext.closePath();
    myContext.fill();
    //在左侧绘制黑色半圆
    myContext.fillStyle="#000";
    myContext.beginPath();
    myContext.arc(200, 200, 150, Math.PI/2, 1.5*Math.PI, false);
    myContext.closePath();
    myContext.fill();
    //在垂直中心上绘制黑、白两个小圆
    myContext.fillStyle="#000";
    myContext.beginPath();
    myContext.arc(200, 275, 75, 0, Math.PI*2, false);
    myContext.closePath();
    myContext.fill();
    myContext.fillStyle="#FFF";
    myContext.beginPath();
    myContext.arc(200, 125, 75, 0, Math.PI*2, false);
    myContext.closePath();
    myContext.fill();
    myContext.fillStyle="#FFF";
    myContext.beginPath();                //绘制白点和黑点
    myContext.arc(200, 275, 10, 0, Math.PI*2, false);
    myContext.closePath();
    myContext.fill();
    myContext.fillStyle="#000";
    myContext.beginPath();
    myContext.arc(200, 125, 10, 0, Math.PI*2, false);
    myContext.closePath();
    myContext.fill();
    myContext.beginPath();                //绘制最外层的空心大圆
    myContext.arc(200, 200, 150, 0, 2*Math.PI, false);
    myContext.closePath();
    myContext.stroke();
  });});
 </script></head>
<body><p><input type="button" value="绘制太极图案" id="myBtn" style="width:350px"></p>
<canvas id="myCanvas" width="410" height="410"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，beginPath()方法用于开始一条路径或重置当前的路径；closePath()方法用于创建从当前点到开始点的路径；arc()方法用于创建弧/曲

线(用于创建圆或部分圆)。整个绘制过程如下：首先绘制两个半圆，分别对应左侧黑色和右侧白色，组成一个圆；绘制完成以后再分别绘制一个白色和黑色的圆，处于绘制好的黑白圆之内，半径恰好是黑白大圆一半；最后在已绘制的两个黑白小圆内分别填充(绘制)白色和黑色的点。

此实例的源文件名是 myHtmlA068.html。

076 在自定义画布上模拟刮刮奖的刮奖特效

此实例主要为画布标签添加 mousedown、mousemove、mouseup 等事件响应方法，并使用 fillRect()方法和 clearRect()方法，从而实现在自定义画布上模拟刮刮奖的刮奖特效。当在 Google Chrome 浏览器中显示该页面时，单击“开始擦除涂层(然后按住鼠标在灰色涂层上移动)”按钮，然后按住鼠标在灰色涂层上移动，即可出现如图 076-1 所示的刮奖效果。有关此实例的主要代码如下。

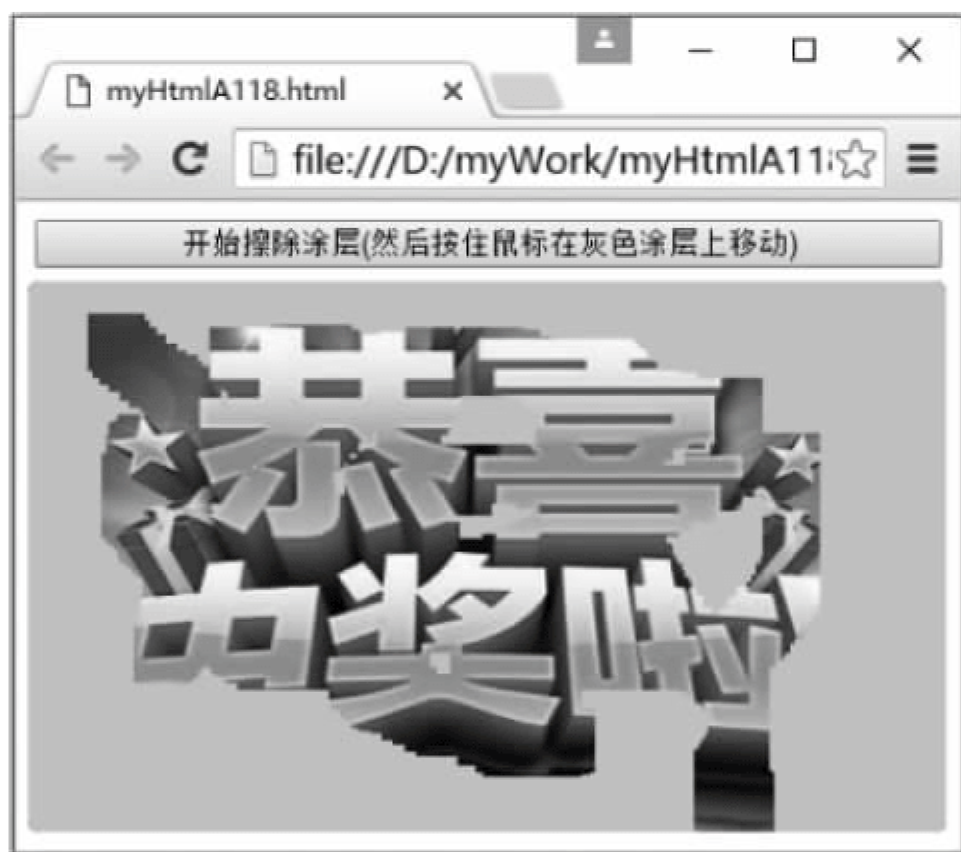


图 076-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script type="text/javascript">
  $(function() {
   var myCanvas = document.getElementById('myCanvas');
   var myContext = myCanvas.getContext('2d');
   myContext.fillStyle = 'lightgrey';        //在画布上增加一个灰色的涂层
   myContext.fillRect(0, 0, 400, 240);
   var myEraser = document.getElementById('myEraser');
   var isErasering;
   myEraser.onclick = function() {           //开始擦除
     myCanvas.onmousedown = function() { isErasering = true; }
     myCanvas.onmousemove = function(e) {    //继续擦除
      if (isErasering) {
       var mouseX = e.pageX - this.offsetLeft;
       var mouseY = e.pageY - this.offsetTop;
       //根据当前位置用边长为25px的正方形擦除涂层
       myContext.clearRect(mouseX, mouseY, 25, 25);
     } }
     myCanvas.onmouseup = function() {       //结束擦除
      isErasering = false;
```

```
    } } });
   </script>
   <style type = "text/css">
    canvas { border - radius: 5px; z - index: 100;
            position: absolute; top: 35px; left: 5px; }
    img { position: absolute; width: 400px; height: 240px;
         border - radius: 5px; z - index: - 1; top: 35px; left: 5px; }
    button { width: 395px; }
   </style></head>
  <body><button id = "myEraser">开始擦除涂层(然后按住鼠标在灰色涂层上移动)</button>
  <img src = "img/a118.jpg"/>
  <canvas width = "400" height = "240" id = "myCanvas"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myContext.fillRect(0, 0, 400, 240)用于在自定义画布上绘制一个实心矩形。在 HTML5 中，fillRect()方法用于绘制“已填色”的矩形，默认的填充颜色是黑色，用户可以使用画布的 fillStyle 属性来设置填充颜色、渐变或模式。fillRect()方法的语法格式如下。

```
fillRect(x,y,width,height);
```

其中，参数 x 表示矩形左上角的 x 坐标；参数 y 表示矩形左上角的 y 坐标；参数 width 表示矩形的宽度，以像素计；参数 height 表示矩形的高度，以像素计。

clearRect(mouseX, mouseY, 25, 25)表示根据鼠标指针的当前位置用边长为 25px 的正方形擦除涂层。在 HTML5 中，clearRect()方法用于清空给定矩形内的指定像素，该方法的语法格式如下。

```
clearRect(x,y,width,height);
```

其中，参数 x 表示要清除的矩形左上角的 x 坐标；参数 y 表示要清除的矩形左上角的 y 坐标；参数 width 表示要清除的矩形的宽度，以像素计；参数 height 表示要清除的矩形的高度，以像素计。

此实例的源文件名是 myHtmlA118.html。

077　绘制局部图像模拟水平或垂直展开图像

此实例主要通过使用定时器方法 setInterval()以及动态设置画布的 drawImage()方法的宽度和高度参数实现模拟水平或垂直展开图像的效果。当在 Google Chrome 浏览器中显示该页面时，单击“沿垂直方向展开图像”按钮，图像将从上向下逐渐展开，如图 077-1 所示；单击“沿水平方向展开图像”按钮，图像将从左向右逐渐展开，如图 077-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
   $ (function() {
    var canvas = $ ('#MyCanvas')[0];
    var context = canvas.getContext('2d');
    var width = canvas.width;
    var height = canvas.height;
    var drawHeight, drawWidth; var timer;
    $ ("#myBtnVertical").click(function() {        //沿垂直方向展开图像
     context.clearRect(0, 0, width, height);
     clearInterval(timer);
```

```
        context.fillStyle = "#FFF";
        context.fillRect(0, 0, width, height);
        drawHeight = 0;
        timer = setInterval(function() {
         context.save();
         //根据指定的高度绘制图像的对应部分
         context.drawImage( $ ("img")[0], 0, 0, width, drawHeight, 0, 0, width, drawHeight);
         drawHeight += 2;
         if (drawHeight > height){                          //结束展开图像动作
          clearInterval(timer); }
         context.restore();
      }, 10);});
      $ ("#myBtnHorizontal").click(function() {           //沿水平方向展开图像
       context.clearRect(0, 0, width, height);
       clearInterval(timer);
       context.fillStyle = "#FFF";
       context.fillRect(0, 0, width, height);
       drawWidth = 0;
       timer = setInterval(function() {
        context.save();
        //根据指定的宽度绘制图像的对应部分
        context.drawImage( $ ("img")[0], 0, 0, drawWidth, height, 0, 0, drawWidth, height);
        drawWidth += 2;
        if (drawWidth > width) {                            //结束展开图像动作
          clearInterval(timer); }
        context.restore(); },10);});});
    </script>
    <style type = "text/css">
     img { display: none; width:400px; height:250px;}
     button { width: 195px; }
    </style></head>
  <body><div align = "center">
    <p><button id = "myBtnVertical">沿垂直方向展开图像</button>
       <button id = "myBtnHorizontal">沿水平方向展开图像</button></p>
    <img src = "img/B099B.jpg"/>
    <canvas id = "MyCanvas" width = "400" height = "250"></canvas></div></body></html>
```



图 077-1



图 077-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，setInterval()函数用于按照指定的周期(10 毫秒)根据指定的宽度和高度执行在画布上绘制图像的代码，该方法会不停地调用参数中指定的代码，直到 clearInterval()函数被调用或窗口被关闭；drawImage($("img")[0], 0, 0, width, drawHeight, 0, 0, width, drawHeight)用于根据指定的高度 drawHeight 获取 $("img")[0]的图像信息，并绘制在画布的对应位置；drawImage($("img")[0], 0, 0, drawWidth, height, 0, 0, drawWidth, height)用于根据指定的宽度 drawWidth，获取 $("img")[0]的图像信息，并绘制在画布的对应位置。

此实例的源文件名是 myHtmlA121.html。

078　绘制局部图像模拟向左、右两端展开图像

此实例主要通过使用 setInterval()、drawImage()、fillRect()方法等模拟从中心向两边展开图像或从两边向中心收缩图像的效果。当在 Google Chrome 浏览器中显示该页面时，单击"从中心向两边展开图像"按钮，图像将从中心逐渐向两边展开，如图 078-1 所示；单击"从两边向中心收缩图像"按钮，图像将从左、右两边同时逐渐向中心收缩，如图 078-2 所示。有关此实例的主要代码如下。



图　078-1



图　078-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var canvas = $('#myCanvas')[0];
   var context = canvas.getContext('2d');
   var width = canvas.width; var height = canvas.height;
   var drawLeft, drawWidth; var timer;
   $("#myBtnSpread").click(function() {                //从中心向两边展开图像
    context.clearRect(0, 0, width, height);
    clearInterval(timer);
    context.fillStyle = "#FFF";
    context.fillRect(0, 0, width, height);
    drawLeft = width/2;
    drawWidth = 0;
    timer = setInterval(function() {
```

```
          context.save();
          //显示指定区域的图像
          context.drawImage( $ ("img")[0], drawLeft, 0, drawWidth, height, drawLeft, 0, drawWidth, height);
          drawLeft -= 1; drawWidth += 2;
          if (drawLeft <= 0) { clearInterval(timer); }
          context.restore(); }, 20); });
      $ ("#myBtnShrink").click(function() {                    //从两边向中心收缩图像
        context.clearRect(0, 0, width, height);
        clearInterval(timer);
        context.fillStyle = "#FFF";
        context.fillRect(0, 0, width, height);
        context.save();
        context.drawImage( $ ("img")[0],0,0,width,height);     //首先显示完整的图像
        context.restore();
        var drawWidth = 0;
        timer = setInterval(function(){
          context.fillStyle = "#FFF";                          //设置填充颜色与背景的颜色相同
          //使用与背景相同的颜色不断遮掩图像的左边
          context.fillRect(0, 0, drawWidth, height);
          //使用与背景相同的颜色不断遮掩图像的右边
          context.fillRect(width, 0, - drawWidth, height);
          drawWidth++;
          if(drawWidth == width/2 + 1){ clearInterval(timer); } },20); });});
    </script>
    <style type = "text/css">
      img { display: none;width:400px;height:250px;}
      button { width: 195px;}
    </style></head>
  <body><div align = "center">
    <p><button id = "myBtnSpread">从中心向两边展开图像</button>
       <button id = "myBtnShrink">从两边向中心收缩图像</button></p>
    <img src = "img/B157A.jpg"/>
    <canvas id = "myCanvas" width = "400" height = "250"></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，setInterval()函数用于按照指定的周期(20毫秒)根据指定的宽度和高度执行在画布上绘制(显示)图像的代码，该方法会不停地调用参数中指定的代码实现绘制动作，直到clearInterval()函数被调用或窗口被关闭；drawImage($("img")[0], drawLeft, 0, drawWidth, height, drawLeft, 0, drawWidth, height)用于根据指定的宽度drawWidth获取$("img")[0]的图像信息，并绘制在画布的对应位置；drawImage($("img")[0], 0, 0, width, height)用于绘制完整的图像；fillRect(0, 0, drawWidth, height)用于以页面背景色不断遮掩图像的左边；fillRect(width, 0, -drawWidth, height)用于以页面背景色不断遮掩图像的右边。

此实例的源文件名是myHtmlA122.html。

079　绘制局部图像模拟以百叶窗风格展开图像

此实例主要通过使用setInterval()、drawImage()等方法模拟以水平(或垂直)百叶窗展开图像的特效。当在Google Chrome浏览器中显示该页面时，单击“以垂直百叶窗风格显示图像”按钮，图像将以百叶窗的风格从上向下展开，如图079-1所示；单击“以水平百叶窗风格显示图像”按钮，图像将以百叶窗的风格从左向右展开，如图079-2所示。有关此实例的主要代码如下。

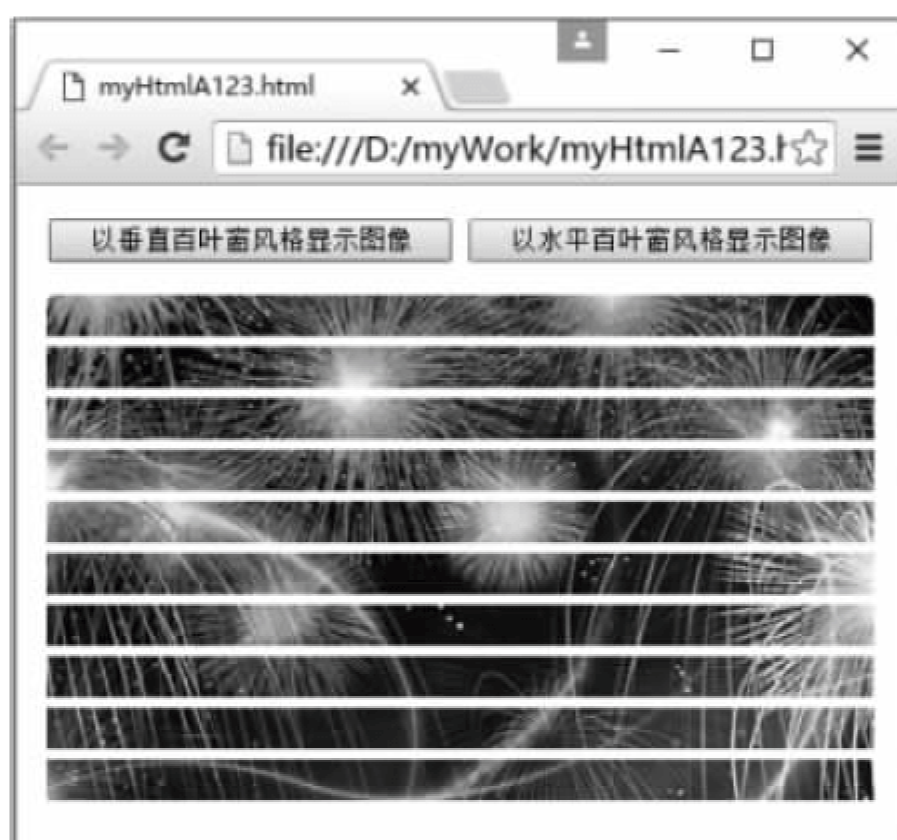


图　079-1

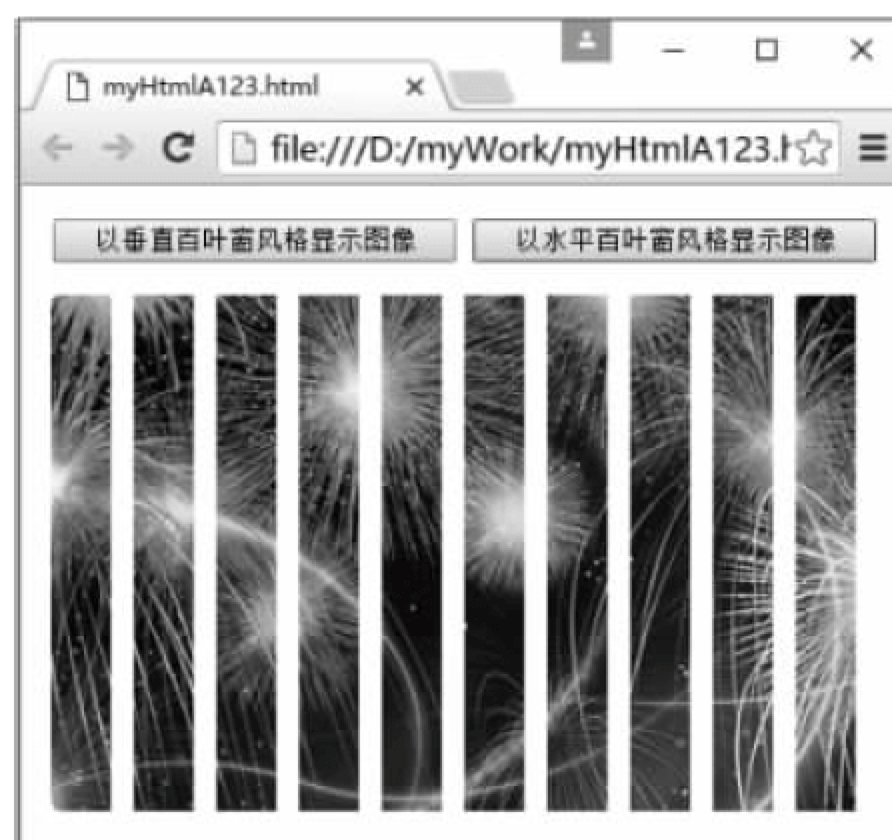


图　079-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   var canvas=$('#myCanvas')[0];
   var context=canvas.getContext('2d');;
   var width=canvas.width;
   var height=canvas.height;
   var drawHeight,drawWidth;
   context.drawImage($("img")[0],0,0);
   var timer;
   $("#myBtnVertical").click(function(){      //以垂直百叶窗风格显示图像
    context.clearRect(0, 0, width, height);
    clearInterval(timer);
    context.fillStyle="#EEEEFF";
    context.fillRect(0, 0,width,height);
    //根据图像的高度分成10份百叶,并获取百叶的高度
    var myHeight = height/10; drawHeight = 0;
    timer=setInterval(function(){             //启动定时器,以百叶窗风格显示图像
     context.save();
     context.clearRect(0,0,width,height);
     for(i=0;i<10;i++){                        //分别绘制10份百叶对应的当前图像
      context.drawImage($("img")[0],0,0 + i * myHeight,width,drawHeight,0,0 + i * myHeight,width,
drawHeight); }
     drawHeight+=1;
     if(drawHeight>myHeight){                  //图像显示完毕,终止定时器
       clearInterval(timer); }
     context.restore(); },20); });
   $("#myBtnHorizontal").click(function(){    //以水平百叶窗风格显示图像
    context.clearRect(0, 0, width, height);
    clearInterval(timer);
    context.fillStyle="#FFF";
    context.fillRect(0, 0,width,height);
    //根据图像的宽度分成10份百叶,并获取百叶的宽度
    var myWidth=width/10; drawWidth=0;
    timer=setInterval(function(){             //启动定时器,以百叶窗风格显示图像
     for(i=0;i<10;i++){                        //分别绘制10份百叶对应的当前图像
      context.drawImage($("img")[0],0 + i * myWidth,0,drawWidth,height, 0 + i * myWidth,0,drawWidth,
height); }
```

```
        drawWidth += 1;
        if(drawWidth > myWidth){          //图像显示完毕,终止定时器
          clearInterval(timer); } },20); });});
  </script>
  <style type = "text/css">
   img { display: none; width:400px; height:250px;}
   button { width: 195px; }
   canvas{ border - radius: 5px;}
  </style></head>
 <body><div align = "center">
  <p><button id = "myBtnVertical">以垂直百叶窗风格显示图像</button>
     <button id = "myBtnHorizontal">以水平百叶窗风格显示图像</button></p>
  <img src = "img/A123.jpg"/>
  <canvas id = "myCanvas" width = "400" height = "250"></canvas><br></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,setInterval()函数用于按照指定的周期(20毫秒)根据指定的百叶窗宽度和高度执行在画布上绘制图像的代码,该方法会不停地调用参数中指定的代码实现绘制动作,直到clearInterval()函数被调用或窗口被关闭;drawImage($("img")[0],0,0+i*myHeight,width,drawHeight,0,0+i*myHeight,width,drawHeight)用于根据指定的百叶窗高度myHeight获取$("img")[0]的图像信息,并绘制在画布的对应位置;drawImage($("img")[0],0+i*myWidth,0,drawWidth,height,0+i*myWidth,0,drawWidth,height)用于根据指定的百叶窗宽度myWidth获取$("img")[0]的图像信息,并绘制在画布的对应位置。

此实例的源文件名是myHtmlA123.html。

080 绘制局部图像模拟向上、下两端展开图像

此实例主要通过使用setInterval()、drawImage()、fillRect()等方法实现从中间向上下展开图像或从上下向中间收缩图像的效果。当在Google Chrome浏览器中显示该页面时,单击"从中间向上下展开图像"按钮,图像将从中间逐渐向上下展开,如图080-1所示;单击"从上下向中间收缩图像"按钮,图像将从上下同时逐渐向中间收缩,如图080-2所示。有关此实例的主要代码如下。



图 080-1

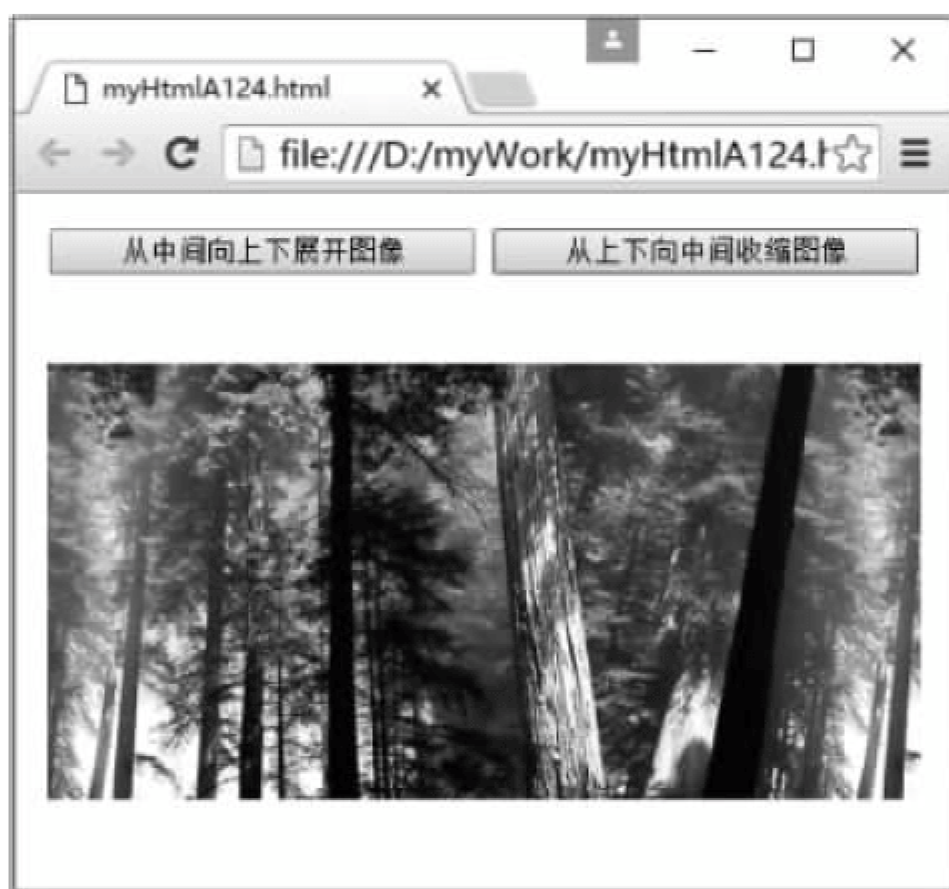


图 080-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var canvas = $('#myCanvas')[0];
   var context = canvas.getContext('2d');
   var width = canvas.width; var height = canvas.height; var timer;
   $("#myBtnSpread").click(function() {                   //从中间向上下展开图像
    context.clearRect(0, 0, width, height);
    clearInterval(timer);
    context.fillStyle = "#FFF";
    context.fillRect(0, 0,width,height);
    var drawTop = height/2; var drawHeight = 0;
    timer = setInterval(function(){
    context.save();
    context.drawImage( $("img")[0],0,drawTop,width,drawHeight,0,drawTop, width,drawHeight);
                                                          //显示指定区域的图像
    drawTop -= 1; drawHeight += 2;
    if(drawTop <= 0){                                     //展开图像完成
     clearInterval(timer); }
    context.restore(); },20); });
  $("#myBtnShrink").click(function() {                    //从上下向中间收缩图像
   context.clearRect(0, 0, width, height);
   clearInterval(timer);
   context.fillStyle = "#FFF";                            //设置填充颜色与背景的颜色相同
   context.fillRect(0, 0,width,height);
   context.drawImage( $("img")[0],0,0,width,height);      //显示完整的图像
   var drawHeight = 0;
   timer = setInterval(function(){
    context.fillRect(0,0,width,drawHeight);
    context.fillRect(0,height,width, - drawHeight);
    drawHeight++;
    if(drawHeight == height/2 + 1){                       //终止遮掩图像
      clearInterval(timer); } },20); }); });
 </script>
 <style type="text/css">
  img { display: none; width: 400px; height: 250px; }
  button { width: 195px;}
  canvas { border-radius: 5px; }
</style></head>
<body><div align="center">
 <p><button id="myBtnSpread">从中间向上下展开图像</button>
    <button id="myBtnShrink">从上下向中间收缩图像</button></p>
 <img src="img/A124.jpg"/>
 <canvas id="myCanvas" width="400" height="250"></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，setInterval()函数用于按照指定的周期(20毫秒)根据指定的宽度和高度执行在画布上绘制图像的代码，该方法会不停地调用参数中指定的代码实现绘制动作，直到clearInterval()函数被调用或窗口被关闭；drawImage($("img")[0],0,drawTop,width,drawHeight,0,drawTop,width, drawHeight)用于根据指定高度drawHeight获取$("img")[0]的图像信息，并绘制在画布的对应位置；drawImage($("img")[0],0,0,width,height)用于绘制完整的图像；fillRect(0,0,width, drawHeight)用于根据页面的背景色绘制矩形不断遮掩图像的上边；fillRect(0,height,width,-drawHeight)用于根据页面的背景色绘制矩形不断遮掩图像的

下边。

此实例的源文件名是 myHtmlA124.html。

081　对图像进行水平拉伸放大或垂直拉伸

此实例主要通过使用 scale()方法实现对图像进行水平拉伸(放大)或垂直拉伸(放大)的效果。当在浏览器中显示该页面时,单击"对图像进行垂直拉伸"按钮,将在下面显示经过垂直拉伸放大的图像,如图 081-1 所示;单击"对图像进行水平拉伸"按钮,将在下面显示经过水平拉伸放大的图像,如图 081-2 所示。有关此实例的主要代码如下。



图　081-1



图　081-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnHorizontal").click(function() {        //对图像进行水平拉伸
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    var myImage=$("#myJpg").get(0);
    myContext.clearRect(0, 0, 410, 410);
    myContext.save();
    myContext.scale(2,1);
    myContext.drawImage(myImage,0,0);
    myContext.restore();
   });
   $("#myBtnVertical").click(function() {          //对图像进行垂直拉伸
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    var myImage=$("#myJpg").get(0);
    myContext.clearRect(0, 0, 410, 410);
    myContext.save();
```

```
        myContext.scale(1,2);
        myContext.drawImage(myImage,0,0);
        myContext.restore(); });});
  </script></head>
 <body><p>原始图像：</p><img id="myJpg" src="img/a067.jpg" />
 <p><input type="button" value="对图像进行垂直拉伸" id="myBtnVertical"
 style="width:170px">
    <input type="button" value="对图像进行水平拉伸" id="myBtnHorizontal"
 style="width:170px"></p>
 <canvas id="myCanvas" width="410" height="410" ></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，save()方法用于保存当前环境的状态；restore()方法用于恢复之前保存过的路径状态和属性；scale()方法用于缩放当前绘图，使其更大或更小。如果对绘图进行缩放，所有之后的绘图也会被缩放。scale()方法的语法声明如下。

```
scale(scalewidth,scaleheight);
```

其中，参数 scalewidth 表示缩放当前绘图的宽度(1=100%，0.5=50%，2=200%，以此类推)；参数 scaleheight 表示缩放当前绘图的高度(1=100%，0.5=50%，2=200%，以此类推)。

此实例的源文件名是 myHtmlA067.html。

082　重新映射画布并按照指定角度旋转图像

此实例主要通过使用 translate()和 rotate()方法实现重新映射画布位置并按照指定角度旋转图像。当在浏览器中显示该页面时，单击“显示经过旋转之后的图像”按钮，将显示从(0,0)位置平移到(120,10)并旋转 30°之后的图像，如图 082-1 所示。有关此实例的主要代码如下。



图　082-1

```
 <!doctype html><html><head><meta charset=UTF-8>
  <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
```

```
$("#myBtn").click(function() {          //显示经过旋转之后的图像
 var myCanvas = $("#myCanvas").get(0);
 var myContext = myCanvas.getContext("2d");
 var myImage = new Image();
 myImage.src = "img/a043.jpg";
 myImage.onload = function() {
 myContext.translate(120,10);          //平移画布
 myContext.rotate(30 * Math.PI/180);   //旋转 30°
 //重新绘制旋转之后的图像
 myContext.drawImage(myImage,0, 0, myImage.width, myImage.height); } }); });
</script></head>
<body><p><input type="button" value="显示经过旋转之后的图像" id="myBtn" style=
"width:300px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，translate()方法用于重新映射画布的(0,0)位置，该方法的语法声明如下。

```
translate(x,y);
```

其中，参数 x 表示平移水平坐标 x 值；参数 y 表示平移垂直坐标 y 值。

rotate()方法用于旋转当前绘制对象，该方法的语法声明如下。

```
rotate(angle);
```

其中，参数 angle 表示旋转角度，以弧度计算。如果需要将角度转换为弧度，应使用 degrees * Math.PI/180 公式进行计算。例如旋转 5°，则使用公式 5 * Math.PI/180。

此实例的源文件名是 myHtmlA043.html。

083　对彩色图像进行水平镜像的特效处理

此实例主要通过数学运算变换像素坐标，从而获取照片的水平镜像效果。当在浏览器中显示该页面时，将在上面显示正常的图像；单击“对图像进行水平镜像特效处理”按钮，将在下面显示经过水平镜像特效处理之后的图像，如图 083-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
 function myEffect(myContext, myData) {               //水平镜像特效
  var myTemp = myContext.createImageData(myData.width, myData.height);
  myTemp.data.set(myData.data);
  for ( var x = 0; x < myTemp.width; x++){
   for ( var y = 0; y < myTemp.height; y++){
    var idx = (x + y * myTemp.width) * 4;             //像素在数组中的索引
    var midx = (((myTemp.width - 1) - x) + y * myTemp.width) * 4;
    //在图像中设置新像素
    myData.data[midx + 0] = myTemp.data[idx + 0];     //R 通道
    myData.data[midx + 1] = myTemp.data[idx + 1];     //G 通道
    myData.data[midx + 2] = myTemp.data[idx + 2];     //B 通道
    myData.data[midx + 3] = 255;                      //alpha 通道
  } } }
 $(document).ready(function() {
```

```
    $("#myBtn").click(function() {            //对图像进行水平镜像特效处理
     var myImage = document.getElementById("myJpg");
     var myCanvas = document.getElementById("myCanvas");
     myCanvas.width = myImage.clientWidth;
     myCanvas.height = myImage.clientHeight;
     myContext = myCanvas.getContext("2d");
     myContext.drawImage(myImage, 0, 0, myCanvas.width, myCanvas.height);
     var myData = myContext.getImageData(0, 0,myCanvas.width, myCanvas.height);
     myEffect(myContext, myData);
     myContext.putImageData(myData, 0, 0); }); });
 </script></head>
<body><p>原始图像:</p><img id="myJpg" src="img/a041.jpg"/>
<p><input type="button" value="对图像进行水平镜像特效处理" id="myBtn" style="width:380px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,getImageData()方法用于获取ImageData对象,该对象复制了画布指定矩形的像素数据ImageData;putImageData()方法用于在画布上显示图像数据(从指定的ImageData对象);createImageData()方法用于创建新的空白ImageData对象。

此实例的源文件名是myHtmlA053.html。



图 083-1

084 给图像添加半透明放射圆形面罩特效

此实例主要通过使用 createRadialGradient()方法给图像添加半透明放射圆形渐变面罩特效。当在浏览器中显示该页面时，单击“给图像添加半透明放射圆形面罩特效”按钮，将在下面显示经过半透明放射圆形渐变面罩处理之后的图像，如图 084-1 所示。有关此实例的主要代码如下。



图 084-1

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {          //给图像添加半透明放射圆形面罩特效
    var myImage=document.createElement('img');
    myImage.src="img/a057.jpg";
    myImage.onload=function() {
    var myCanvas=document.getElementById("myCanvas");
    var myContext=myCanvas.getContext("2d");
    myContext.beginPath();                  //把矩形照片裁剪成一个圆
    myContext.arc(205, 205, 205, 0, 2*Math.PI);
    myContext.closePath();
    myContext.clip();
    myContext.drawImage(myImage, 0, 0, 410, 410);
    //添加半透明放射圆形面罩
    var myGradient=myContext.createRadialGradient(myCanvas.width/2,
        myCanvas.height/2, 10, myCanvas.width/2,myCanvas.height/2, 200);
    myGradient.addColorStop(0, 'rgba(247, 247, 247, 0)');
    myGradient.addColorStop(0.7, 'rgba(120, 120, 120, 0.5)');
    myGradient.addColorStop(0.9, 'rgba(0, 0, 0, 0.8)');
```

```
        myGradient.addColorStop(1, 'rgba(238, 238, 238, 1)');
        myContext.beginPath();
        myContext.arc(myCanvas.width/2,myCanvas.height/2, 300, 0, Math.PI * 2, true);
        myContext.closePath();
        myContext.fillStyle = myGradient;
        myContext.fill();
      } });});
    </script></head>
  <body><p><input type = "button" value = "给图像添加半透明放射圆形面罩特效" id = "myBtn" style =
  "width:410px"></p>
  <canvas id = "myCanvas" width = "410" height = "410"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，createRadialGradient()方法用于创建放射状/圆形渐变对象，该方法的语法声明如下。

```
createRadialGradient(x0,y0,r0,x1,y1,r1)
```

其中，参数 x0 表示渐变的开始圆的 x 坐标；参数 y0 表示渐变的开始圆的 y 坐标；参数 r0 表示开始圆的半径；参数 x1 表示渐变的结束圆的 x 坐标；参数 y1 表示渐变的结束圆的 y 坐标；参数 r1 表示结束圆的半径。

createRadialGradient()方法的返回值是一个 gradient 对象。当 gradient 对象创建成功之后，需要使用该对象的 addColorStop()方法规定 gradient 对象中的颜色和位置。如果不对 gradient 对象使用该方法，那么渐变将不可见。addColorStop()方法的语法声明如下：

```
addColorStop(stop,color)
```

其中，参数 stop 的值为 0.0～1.0，表示渐变开始与结束之间的位置；参数 color 表示在结束位置显示的 CSS 颜色值。

此实例的源文件名是 myHtmlA057.html。

085　设置填充样式以平铺的风格显示图像

此实例主要通过使用 createPattern()方法实现在矩形中平铺图像。当在浏览器中显示该页面时，单击“平铺图像”按钮，将在下面的矩形中平铺多个图像，如图 085-1 所示。有关此实例的主要代码如下。



图　085-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {              //平铺图像
    var myCanvas=$("#myCanvas").get(0);
    var myContext=myCanvas.getContext("2d");
    var myImage=new Image();
    myImage.src="img/a045.jpg";
    myImage.onload=function() {                 //创建平铺填充样式
     var myPattern=myContext.createPattern(myImage, 'repeat');
     myContext.fillStyle=myPattern;             //设置填充样式
     myContext.fillRect(0, 0, 400, 210);        //填充矩形
  }});});
 </script></head>
<body><p><input type="button" value="平铺图像" id="myBtn" style="width:400px"></p>
<canvas id="myCanvas" width="900" height="950"></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，createPattern()方法用于在指定的方向内重复指定的元素，这些元素可以是图片、视频或者其他 canvas 元素，被重复的元素可用于绘制/填充矩形、圆形或线条等。createPattern()方法的语法声明如下。

```
createPattern(image,"repeat|repeat-x|repeat-y|no-repeat");
```

其中，参数 image 规定要使用的图片、画布或视频元素。第二个参数是重复模式，其中 repeat 是默认值，该模式表示在水平和垂直方向重复；repeat-x 模式表示只在水平方向重复；repeat-y 模式表示只在垂直方向重复；no-repeat 模式表示只显示一次（不重复）。

此实例的源文件名是 myHtmlA045.html。

086 将画布的内容保存为 png 格式的文件

此实例主要在 toDataURL()方法中设置 image/png 参数，从而实现把画布上显示的所有内容保存为本地 png 图像文件。当在 Google Chrome 浏览器中显示该页面时，即可在红线指定的画布范围内任意涂鸦，如图 086-1 所示，单击“清除画布内容”按钮，则将清除涂鸦内容；单击“保存画布内容”按钮，则涂鸦内容将被保存为本地 png 图像文件，在“360 看图”中显示涂鸦 png 图像文件的效果如图 086-2 所示。有关此实例的主要代码如下。

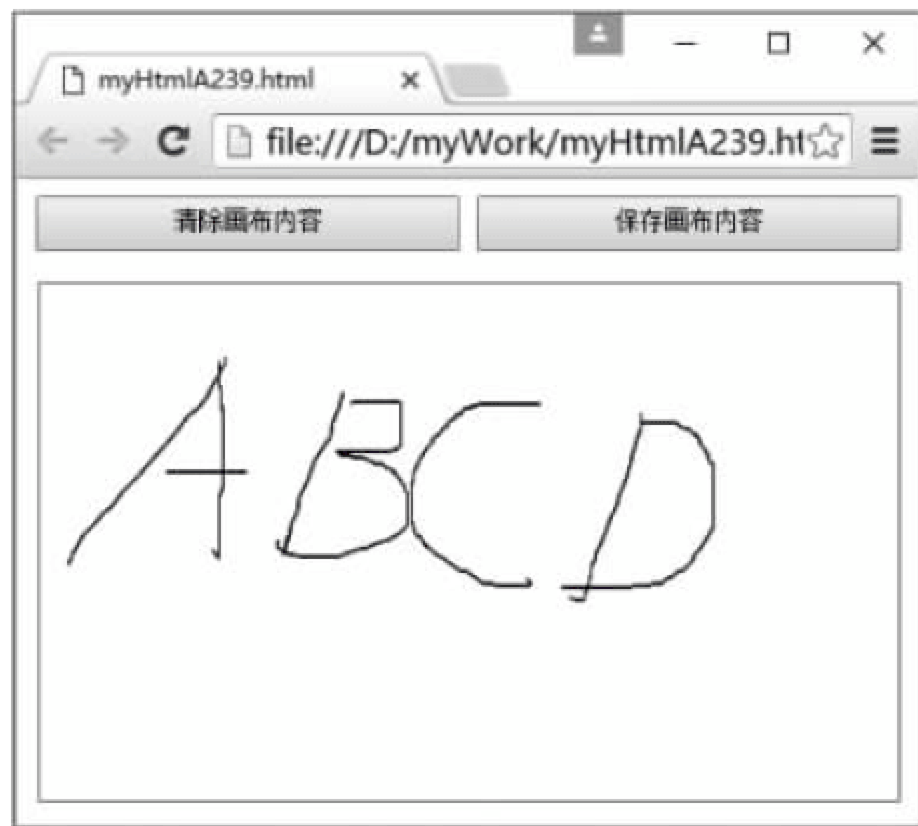


图 086-1



图 086-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   function convertCanvasToImage(canvas) {          //将 canvas 画布内容转换为 png 图片
    var image = new Image();
    image.src = canvas.toDataURL("image/png");
    return image; }
   var context = $("#myCanvas")[0].getContext("2d");
   var LastX, LastY, CurrentX, CurrentY; var isDraw = false;
   $("#myCanvas").on("mousedown", function(e) {
    CurrentX = LastX = e.pageX - $(this).offset().left;
    CurrentY = LastY = e.pageY - $(this).offset().top;
    isDraw = true; });
   $("#myCanvas").on("mousemove", function(e) {
    if (isDraw) {
     $("#myCanvas").css("cursor", "default");
     CurrentX = e.pageX - $(this).offset().left;
     CurrentY = e.pageY - $(this).offset().top;
     context.beginPath();
     context.moveTo(CurrentX, CurrentY);
     context.lineTo(LastX, LastY);
     context.closePath();
     context.strokeStyle = $("input[type=color]").val();
     context.stroke();
     LastX = CurrentX; LastY = CurrentY; } });
  $("#myCanvas").on("mouseup", function(e) { isDraw = false; });
  $("#myClear").click(function() {       //清除画布内容
   context.clearRect(0, 0, 300, 200); });
  $("#mySave").click(function() {        //保存画布内容
   var image = convertCanvasToImage($("canvas")[0]);
   $(this).prop("href", image.src); });});
 </script>
 <style type="text/css">
  a { -webkit-appearance: button; text-decoration: none; height: 17px;
      font-size: 12px; padding: 5px; cursor: default; width:195px;}
  a:link { color: black;}
  canvas { margin-top: 15px; border: 1px solid red; }
</style></head>
<body><div align="center"><a id="myClear">清除画布内容</a>
 <a id="mySave" download="canvas">保存画布内容</a></div>
<div align="center"><canvas id="myCanvas" width="415" height="250"></canvas> </div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，toDataURL("image/png")用于将画布上显示的所有内容保存为本地 png 图像文件。测试表明，当设置该方法为 toDataURL()时，同样是将画布上显示的所有内容保存为本地 png 图像文件。

此实例的源文件名是 myHtmlA239.html。

087 将画布的内容保存为 jpeg 格式的文件

此实例主要通过使用 toDataURL()方法实现把画布上显示的图像、文字等所有内容保存为一幅图像。当在浏览器中显示该页面时，单击“在画布上显示图像和文字”按钮，将在画布上显示一幅图像

和一行文字，如图087-1所示；单击“在新窗口中显示保存结果”按钮，则浏览器将打开一个新窗口显示包含图像和文字的图像，使用鼠标右键单击该图像，可以在滑出的右键菜单中选择“图片另存为(V)…”命令将该图像保存到本地计算机中，如图087-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnShow").click(function() {                //在画布上显示图像和文字
    var myCanvas = document.getElementById("myCanvas");
    var myContext = myCanvas.getContext('2d');
    var myImage = new Image();
    myImage.src = "img/a040.jpg";
    myImage.onload = function() {                    //绘制图像
     myContext.drawImage(myImage,0, 0,myCanvas.width,myCanvas.height);
     myContext.font = '16pt 宋体';
     myContext.lineWidth = 3;
     myContext.strokeStyle = 'green';                //设置描边颜色
     myContext.strokeText('希拉里.克林顿', 140, 50);  //绘制描边文本
     myContext.fillStyle = 'white';                  //设置填充颜色
     myContext.fillText('希拉里.克林顿', 140, 50);    //绘制填充文本
    } });
   $("#myBtnSave").click(function() {                //在新窗口中显示保存结果
    var myCanvas = document.getElementById("myCanvas");
    var myImage = myCanvas.toDataURL("image/jpeg");
    var myWindow = window.open('about:blank','image from canvas');
    myWindow.document.write("<img src='" + myImage + "'alt='from canvas'/>"); });});
 </script></head>
<body><p><input type="button" value="在画布上显示图像和文字" id="myBtnShow" style="width:
410px"></p>
<p><input type="button" value="在新窗口中显示保存结果" id="myBtnSave" style="width:410px"></p>
<canvas id="myCanvas" style="width:410px;height:200px" ></canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，toDataURL(type, ratio)方法用于保存图像，参数type的值可以在image/png、image/jpeg、image/svg+xml等MIME类型中选择(可以不填，默认是image/png)。如果type是“image/jpeg”，可以有第二个参数；如果第二个参数ratio的值在0到1之间，则表示JPEG的质量等级，否则使用浏览器内置的默认质量等级。

此实例的源文件名是myHtmlA064.html。



图 087-1



图 087-2

088　读取并显示沙箱系统中的图像文件

此实例主要通过解析文件对象的 URL 实现在网页中显示本地沙箱文件系统中的图像文件。为了达到更加明晰的测试效果，首先在 Google Chrome 浏览器中打开源代码中的文件 myHtmlA078.html，单击“选择文件”按钮，在弹出的“打开”对话框中选择一个图像文件，例如 a031.jpg，如图 088-1 所示，单击“打开(O)”按钮，则刚才选择的图像文件 a031.jpg 将从本地计算机复制到沙箱系统中，如图 088-2 所示。然后在 Google Chrome 浏览器中打开实例程序的源代码文件 myHtmlA086.html，在“图像文件名称:”文本框中输入刚才复制的文件 a031.jpg，单击“显示图像文件”按钮，将在下面显示此图像文件和完整的 URL 路径，如图 088-3 所示。有关此实例的主要代码如下。

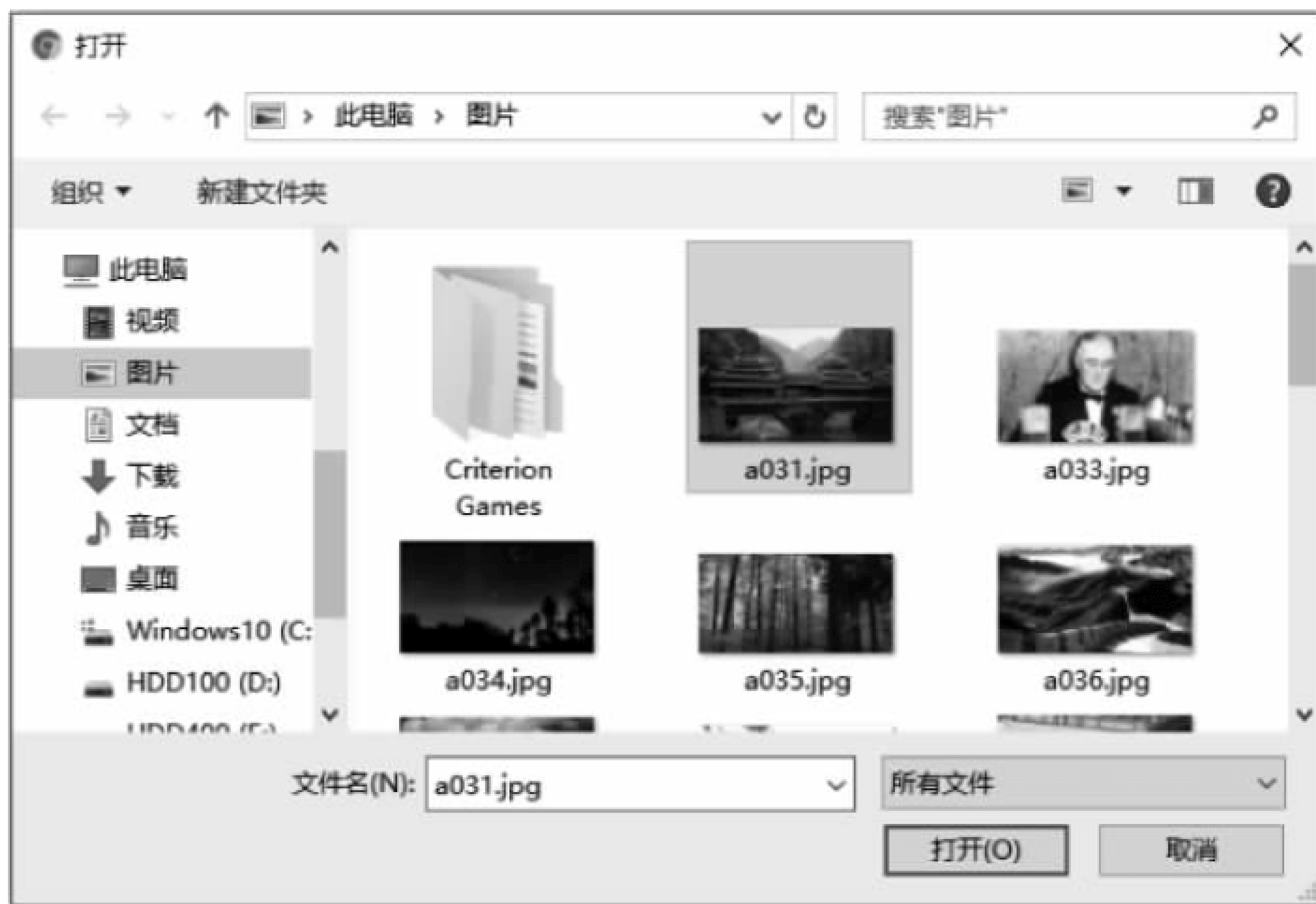


图　088-1

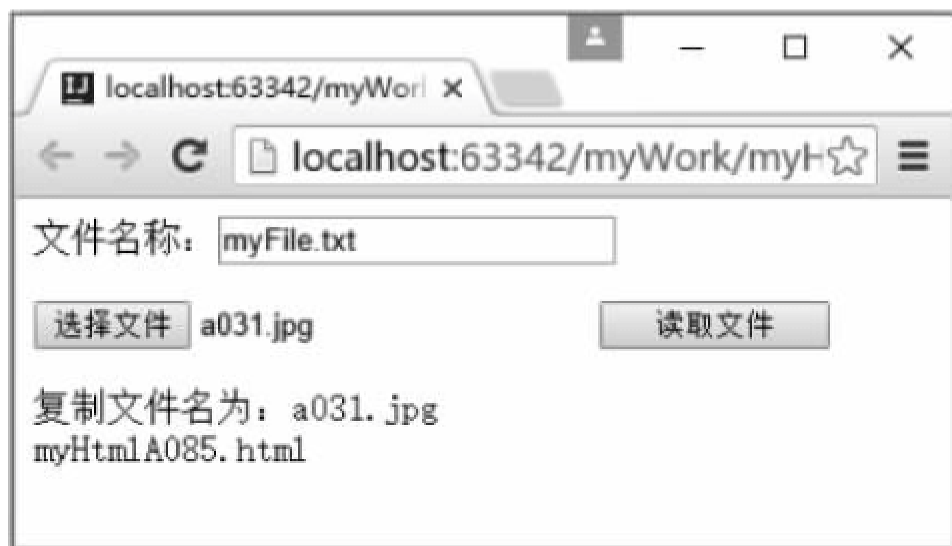


图　088-2



图　088-3

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  window.requestFileSystem =
                 window.requestFileSystem || window.webkitRequestFileSystem;
  function showImage(){                        //显示图像文件
   window.webkitRequestFileSystem(PERSISTENT,1024,
    function(fs){                              //请求文件系统成功时所执行的回调函数
     var myFile = document.getElementById("myFile").value;
     //获取文件对象
     fs.root.getFile(myFile,{create:false},
      function(fileEntry) {                    //获取文件成功时所执行的回调函数
       var myImage = document.createElement('img');
       var myURL = fileEntry.toURL();
       myImage.src = myURL;
       $("p").get(0).appendChild(myImage);
       document.getElementById("myInfo").innerHTML = myURL;
     },errorHandler); },errorHandler); }
  function errorHandler(e){     //请求文件系统失败时所执行的回调函数
   switch (e.code) {
    case FileError.QUOTA_EXCEEDED_ERR:
      msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
      break;
     case FileError.NOT_FOUND_ERR:
      msg = '未找到文件或目录';
      break;
     case FileError.SECURITY_ERR:
      msg = '操作不当引起安全性错误';
      break;
     case FileError.INVALID_MODIFICATION_ERR:
      msg = '对文件或目录所指定的操作不能被执行';
      break;
     case FileError.INVALID_STATE_ERR:
      msg = '指定的状态无效';
    };
 document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
 }
</script></head>
<body><p>图像文件名称: <input type="text" id="myFile" style="width:135px">
<input type="button" value="显示图像文件" onclick="showImage()"
 style="width:120px"><br><br></p><output id="myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，fileEntry.toURL()用于解析文件对象的 URL。在 FileSystem API 中可以使用带有“filesystem:”前缀的 URL，这种 URL 通常用在页面上元素的 href 属性值或 src 属性值中。例如，沙箱中有一个被永久保存的图像文件 a031.jpg，如果需要在页面上显示此图像文件，可以将 img 元素的 src 属性值设置为“filesystem:http://”＋应用程序域名＋“/persistent(当图像文件被保存在永久存储空间时)或 temporary(当图像文件被保存在临时存储空间时)/”＋“a031.jpg”。

此实例的源文件名是 myHtmlA086.html。

089 以二进制形式读取并显示本地图像文件

此实例主要使用 FileReader 对象的 readAsBinaryString()方法和 btoa()方法以二进制形式读取本地图像文件并显示在页面中。当在 Google Chrome 浏览器中显示该页面时，单击“选择文件”按钮，在弹出的“打开”对话框中选择一个图像文件，例如 a038.jpg，如图 089-1 所示，单击“打开(O)”按钮返回页面；再单击“显示图像”按钮，将在下面显示此图像文件，如图 089-2 所示。有关此实例的主要代码如下。

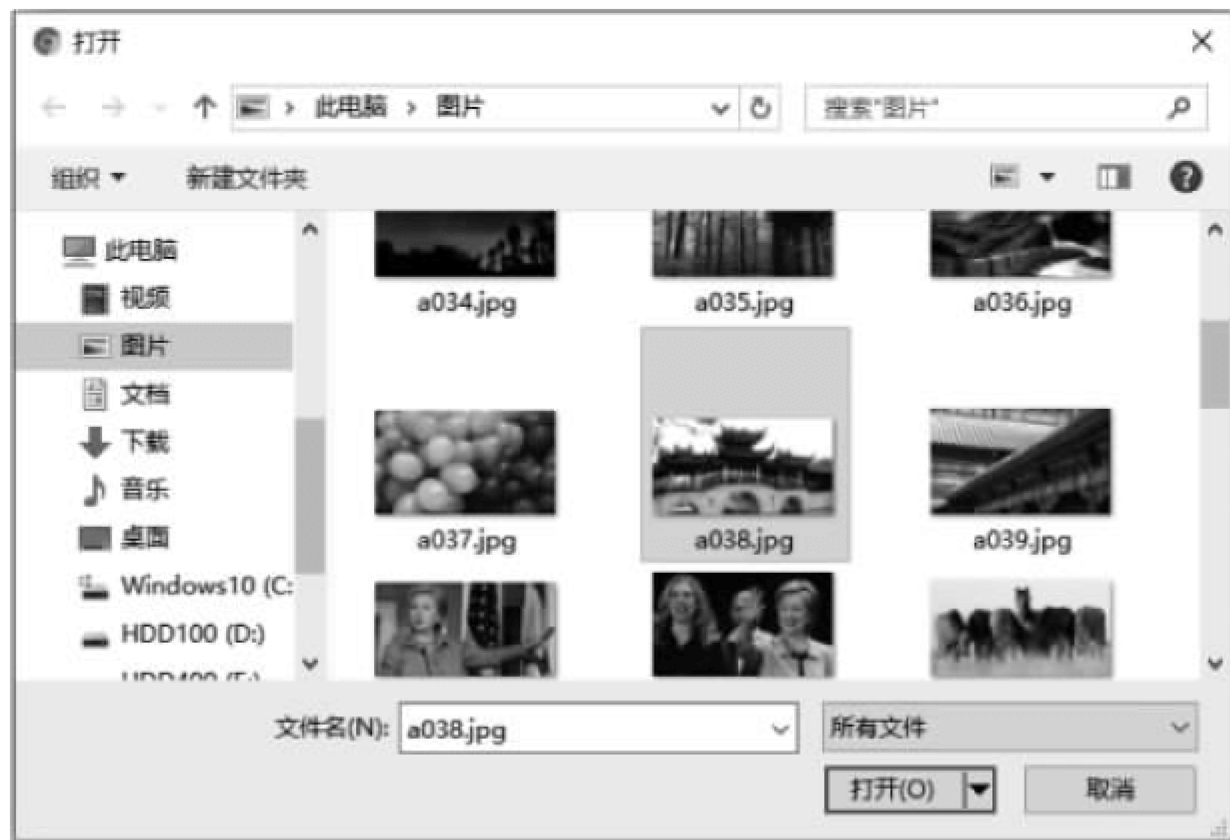


图 089-1



图 089-2

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function showImage(){                        //显示图像
   var myFile=document.getElementById("myFile").files[0];
   if(!/image\/\w+/.test(myFile.type)){        //检查是否为图像文件
     alert("请确保文件为图像类型");
     return false; }
   var myReader=new FileReader();
   myReader.readAsBinaryString(myFile);        //以二进制形式读取图像文件
   myReader.onload=function(f){
    var myImage=document.getElementById("myImage");
    var mySrc="data:"+myFile.type+";base64,"+window.btoa(this.result);
    myImage.innerHTML='<img src="'+mySrc+'"/>' } }
 </script></head>
<body><p><input type="file" id="myFile" style="width: 250px"/>
   <input type="button" value="显示图像" onclick="showImage()" style="width: 120px" /></p>
  <div id="myImage"></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，FileReader 对象的 readAsBinaryString()方法用于读取图像文件的二进制数据；btoa()方法用于将一个字符串进行 Base64 编码处理，该方法使用一个参数，参数值是一串由二进制数据组成的 Unicode 字符串，返回值即是编码后的 Base64 格式的字符串。然后只需将页面上 img 元素的 src 属性值指定为“data:”+用户选择图像文件的 type 属性值+“;base64,”+编码后的 Base64 格式的字符串即可。

此实例的源文件名是 myHtmlA087.html。

090 从本地计算机中选择图像文件并全屏显示

此实例主要使用 readAsBinaryString()方法和 requestFullscreen()方法以二进制形式读取本地计算机中保存的图像文件并在屏幕上全屏显示。当在 Google Chrome 浏览器中显示该页面时，单击"选择文件"按钮，在弹出的"打开"对话框中选择一个图像文件，例如 a106.jpg，如图 090-1 所示，单击"打开(O)"按钮返回页面，该图像将以原始大小显示在页面中；单击"全屏显示"按钮，将在屏幕上全屏显示该图像，如图 090-2 所示。有关此实例的主要代码如下。



图 090-1



图 090-2

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function showImage() {                        //当从本地计算机中选择文件以后显示图像
   var myFile=document.getElementById("myFile").files[0];
   if(!/image\/\w+/.test(myFile.type)){         //检查是否为图像文件
     alert("请确保文件为图像类型");
       return false; }
   var myReader=new FileReader();
   myReader.readAsBinaryString(myFile);         //以二进制形式读取图像文件
   myReader.onload=function(f){
   var myDiv=document.getElementById("myDiv");
   var mySrc="data:"+myFile.type+";base64,"+window.btoa(this.result);
   myDiv.innerHTML='<img id="myImage" src="'+mySrc+'"/>' } }
 function fullScreen() {                        //全屏显示
  myImage=document.getElementById("myImage");
  //将 myImage 设为全屏状态
  myImage.requestFullscreen=myImage.requestFullscreen ||
                         myImage.mozRequestFullscreen || myImage.mozRequestFullScreen ||
                         myImage.webkitRequestFullscreen;
  myImage.requestFullscreen(); }
 </script></head>
<body><p>
<input type="file" id="myFile" style="width:263px" onchange="showImage()"/>
<input type="button" value="全屏显示" onclick="fullScreen()" style="width: 120px"/></p>
<div id="myDiv"></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，FileReader 对象的 readAsBinaryString()方法用于读取图像文件的二进制数据，文件一旦开始读取，无论成功或失败，实例的 result 属性都会被填充。如果读取失败，则 result 值为 null，否则即是读取文件的数据，绝大多数的程序都会在成功读取文件的时候抓取这个值。<img >元素的 RequestFullScreen(谷歌的 Chrome 浏览器是 webkitRequestFullScreen)属性值用于判断浏览器是否支持 FullScreen API，如果支持，则可以直接调用该元素的 requestFullScreen()方法实现将<img >元素设置为全屏显示状态。

此实例的源文件名是 myHtmlA106.html。

091　将本地图像文件或文本文件拖放到网页中

此实例主要通过解析 dataTransfer 对象的文件信息将本地图像文件或文本文件拖放到网页中显示。在 Google Chrome 浏览器中显示该页面时，使用鼠标从本地计算机中拖动一个图像文件到网页中，如图 091-1 所示；然后释放鼠标，该图像文件将显示在网页中，如图 091-2 所示。有关此实例的主要代码如下。

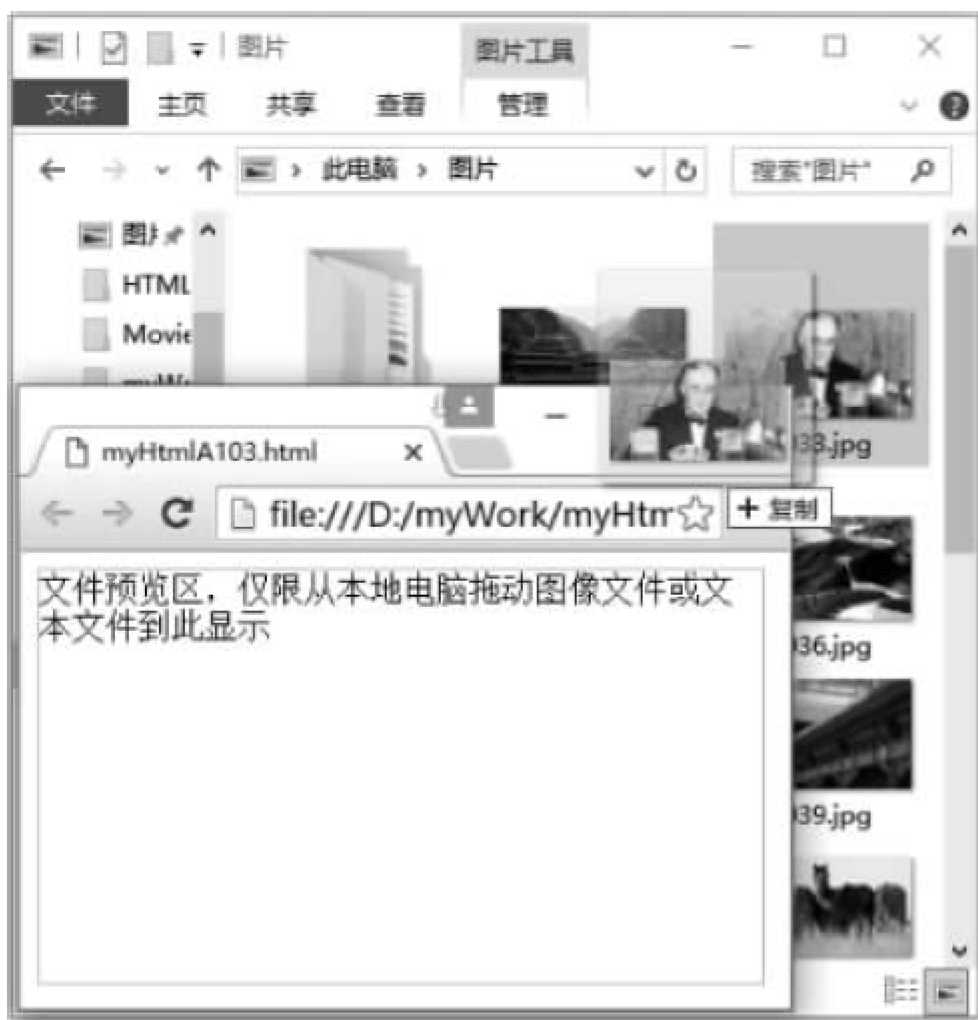


图　091-1

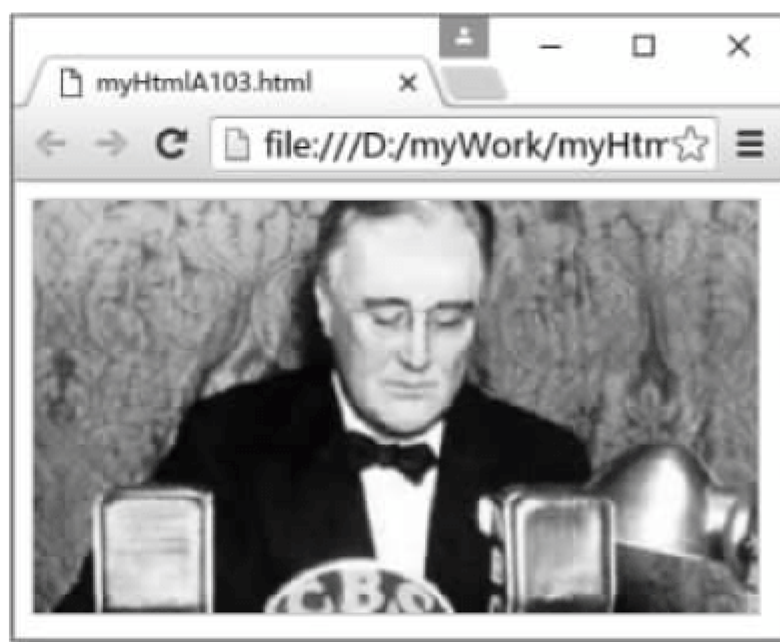


图　091-2

```
<!doctype html><html><head><meta charset = UTF - 8>
<script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function init() {
   var myTarget = document.getElementById("myTarget");
   myTarget.addEventListener("dragover", function(myEvent) {     //拖放文件滑过应用
     myEvent.stopPropagation();
     myEvent.preventDefault();
     }, false);
   myTarget.addEventListener("dragend", function(myEvent) {      //拖放文件结束
    myEvent.stopPropagation();
    myEvent.preventDefault(); }, false);
    //拖放文件结束，并释放鼠标
   myTarget.addEventListener("drop", function(myEvent) {
    myEvent.stopPropagation();
```

```
        myEvent.preventDefault();
        var myFile = myEvent.dataTransfer.files[0];
        var myReader = new FileReader();
        if(/image\/\w+/.test(myFile.type)){
         myReader.readAsDataURL(myFile);
         myReader.onload = function(myEvent) {
         myTarget.style.background =
                   'url(' + myEvent.target.result + ') no-repeat center';
         myTarget.innerHTML = "";
        };} else if (myFile.type.substr(0, 4) == "text") {
         myReader.readAsText(myFile);
         myReader.onload = function(f) {
           $("#myTarget").text(this.result);
        } } else {
         myTarget.innerHTML = "暂不支持此类文件的预览";
         myTarget.style.background = "white";
        } },false); }
        //设置页面属性,不执行默认处理(拒绝拖放)
        document.ondragover = function(myEvent) {
         myEvent.preventDefault(); };
        document.ondrop = function(myEvent) {
         myEvent.preventDefault(); }
        window.onload = init;
   </script></head>
  <body><div id="myTarget" style="border: 1px solid #CCC; width: 350px;height: 200px;"> 文件预览区,
  仅限从本地电脑拖动图像文件或文本文件到此显示</div>
  </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,dataTransfer.files 表示在进行拖放操作时传递的文件对象。该文件对象可以使用 FileReader 对象的成员方法进行解析。FileReader 对象的 readAsDataURL()方法用于将文件解析为 DataURL,即一段以 data:开头的字符串,DataURL 是一种将小文件直接嵌入文档的方案,这里的小文件通常是指图像与 html 等格式的文件。FileReader 对象的 readAsText()用于将文件解析为纯文本,该方法有两个参数,其中第二个参数是文本的编码方式,默认值为 UTF-8。这个方法非常容易理解,它将文件以文本方式读取,读取的结果即是这个文本文件的内容。FileReader 对象的 readAsBinaryString()方法用于将文件读取为二进制字符串,通常将它传送到后台,后台可以通过这段字符串存储为文件。

此实例的源文件名是 myHtmlA103.html。

092　通过超链接的 download 属性下载图片

此实例主要通过设置超链接的 download 属性实现单击超链接(图片)即可下载该图片的功能。当在浏览器中显示该页面时,单击图片将立即下载此图片,如图 092-1 所示。有关此实例的主要代码如下。

```
  <!doctype html><html><head><meta charset = UTF-8></head>
  <body><figure><a href = "#" download = "img/a027.jpg"><img src = "img/a027.jpg" width = "350" height
   = "200"/></a><figcaption>单击此图片即可实现下载</figcaption></figure></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,超链接 a 标签的 download 属性是 HTML5 新增的属性。download 属性是 HTML5 规范的一部分,它表现为一个下载链接,而不是一个导航的链接。download 属性也允许重命名一个需要下载的文件,例如一个文件存放在服务器上,



图　092-1

如果这个文件是自动生成的，一般来说它被都命名为一个系统的数字和破折号的组合，如 acme-doc-2.0.1.txt，可以重命名这个下载文件的名字，用户下载后看到的文件名可以是一个比较好的名字，如 Acme Documentation (ver. 2.0.1).txt，以增强用户体验。

此实例的源文件名是 myHtmlA027.html。

093　定制个性化的虚线作为图像的边框线

此实例主要通过使用 setLineDash()方法产生断点效果，从而定制个性化的虚线作为图像的边框线。当在 Google Chrome 浏览器中显示该页面时，单击"无边框图像"按钮，则显示的图像没有边框线；单击"虚线边框图像"按钮，则在图像的周围将显示定制的虚线边框，如图 093-1 所示。有关此实例的主要代码如下。



图　093-1

```
<!doctype html><html><head><meta charset="UTF-8">
<script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
```

```
    var context = $ ("#canvas")[0].getContext("2d");
    $ ("#myBtnBorder").click(function(){       //虚线边框图像
     context.drawImage( $ ("img")[0], 18, 18, 400, 250);
     context.setLineDash([1,12,23]);
     context.strokeStyle = "blue"; context.lineWidth = 5;
     context.moveTo(14, 14); context.lineTo(423, 14);
     context.lineTo(423, 272); context.lineTo(14, 272);
     context.lineTo(14, 14); context.stroke();
   });
   $ ("#myBtnDefault").click(function() {       //无边框图像
     context.clearRect(0, 0, 430, 280);
     context.drawImage( $ ("img")[0], 18, 18, 400, 250); });});
 </script>
 <style type = "text/css">
  img { width: 400px; height: 250px; display: none;}
  button{ width:203px; }
 </style></head>
<body><div align = "center">
 <p><button id = "myBtnBorder">虚线边框图像</button>
    <button id = "myBtnDefault">无边框图像</button></p>
 <img src = "img/A126.jpg"/><canvas id = "canvas" width = "440" height = "280"></canvas></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，setLineDash([1,12,23])方法用于设置当前 Line 的缓冲效果，即产生断点效果，该方法的参数是一个数字或由一段数字组成的数组。

此实例的源文件名是 myHtmlA126.html。

094　以拖动图像边框线的方式实现图像缩放

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 中的 resizable()方法以拖动图像边框线的方式放大或缩小图像。当在 Google Chrome 浏览器中显示该页面时，使用鼠标拖动图像的右边框线则水平放大图像，如图 094-1 所示；使用鼠标拖动图像的下边框线则垂直放大图像，如图 094-2 所示；在右下角使用鼠标拖动图像，则同时在水平方向和垂直方向上放大(或缩小)图像。有关此实例的主要代码如下。



图　094-1



图　094-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script>
 <script type="text/javascript">
  $(function() {          //拖放图像的边框线改变图像的大小
   $(".myImage").resizable({containment: $(".myContainer"),animate:true });
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   .myContainer { width: 100%; height: 1500px;}
   .myImage { width: 150px; height: 150px;background: url("img/A129.jpg");
              background-size: 100% 100%; border-radius: 5px; }
 </style></head>
<body><div class="myContainer" align="center">
    <div class="myImage"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(".myImage").resizable({containment：$(".myContainer"),animate:true })用于以动画的形式缩放图像。resizable()方法常用的参数如下。

(1) alsoResize:false：类型为 string，可以伴随缩放的元素，如果要在 $("#win")缩放的同时"#win-1"也进行缩放，则可以传"#win-1"。

(2) animate:false：类型为 boolean，缩放是否带动画效果，默认为 false。

(3) delay:null：类型为 number，缩放延迟。

(4) animateDuration:"slow"：类型为 string，鼠标松开时缩放的动画效果。

(5) animateEasing:"swing"：类型为 string，鼠标拖动时的效果。

(6) aspectRatio:false：类型为 number，缩放比例，可以形如 16/9 传参数，默认为 false。

(7) autoHide:false：类型为 boolean，是否隐藏右下角的缩放块，true 为隐藏，在缩放的时候显示。

(8) containment:false：类型为 string，限制缩放的范围，例如传"#win-area"，则只能在 id 为'win-area'的元素里缩放，默认为 false。

(9) ghost:false：类型为 boolean，阴影缩放，待松开鼠标时才缩放到阴影大小的位置。

(10) grid:false：类型为 number，缩放单位，例如 grid:50，默认为 false。

(11) handles:"e,s,se"：类型为 string，缩放时的方向，默认为'e,s,se'，即可以向东、南、东南 3 个方向缩放，可以改为'n,e,s,w,se,sw,ne,nw'，即向各个方向都可以缩放。

(12) helper:false：类型为 string，缩放边框样式，例如 helper："ui-resizable-helper"，默认为 false。

(13) maxHeight:null：类型为 number，缩放时的最大高度，默认为 null。

(14) maxWidth:null：类型为 number，缩放时的最大宽度，默认为 null。

(15) minHeight:10：类型为 number，缩放时的最小高度，默认为 10。

(16) minWidth:10：类型为 number，缩放时的最小宽度，默认为 10。

(17) zIndex:IE3：类型为 number，缩放时的最大高度，默认为 null。

需要说明的是，在使用 resizable()方法缩放图像前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA129.html。

095 以拖动方式将图像移动到页面中的任意位置

此实例主要使用 jquery-ui. min. js 和 jquery-ui. min. css 中的 draggable()方法，从而实现拖动图像即可移动图像到页面中的任意位置。当在 Google Chrome 浏览器中显示该页面时，小兔子位于页面的左上角，如图 095-1 所示；使用鼠标拖动小兔子到页面中的任意位置，例如右下角，松开鼠标之后小兔子停留在右下角，如图 095-2 所示。有关此实例的主要代码如下。

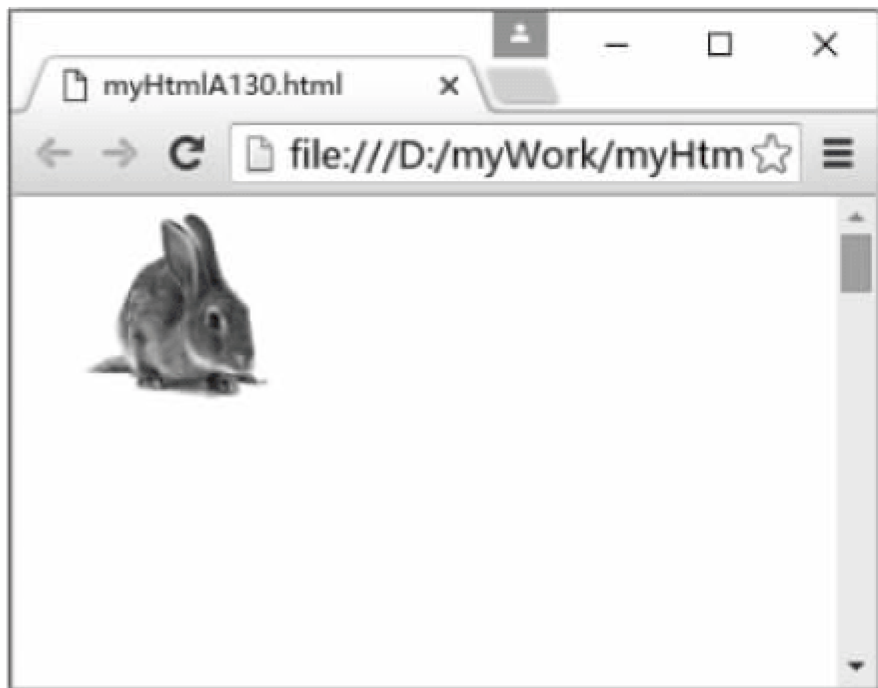


图 095-1

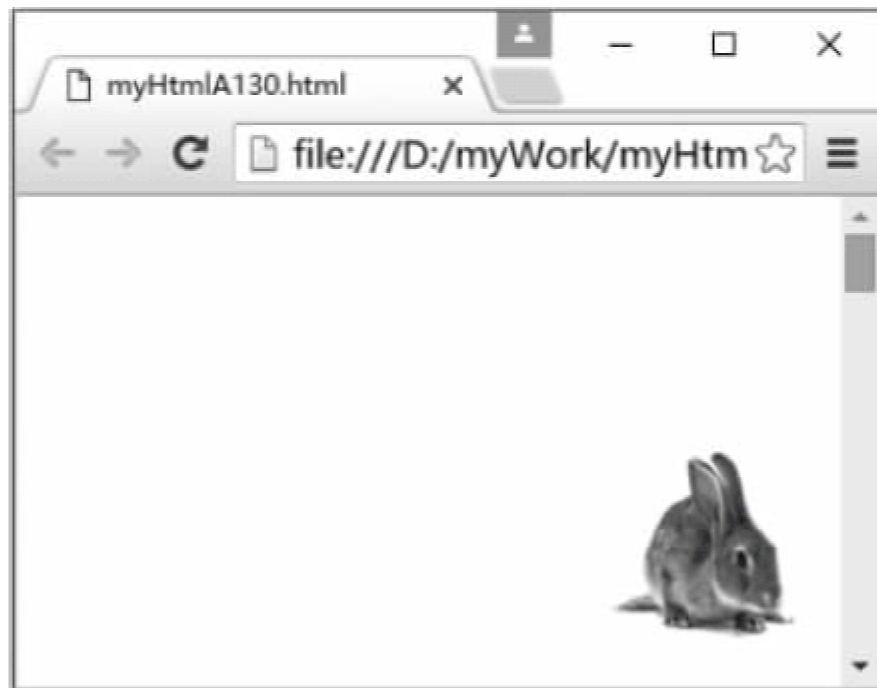


图 095-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function(){       //拖动图像可移动图像到页面中的任意位置
   $(".myImage").draggable({containment: $(".myContainer"),cursor: "move"});});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  .myContainer { width: 100%; height: 1500px;}
  .myImage {width: 80px;height: 80px;background: url("img/A130.png");
          background-size: 100% 100%;}
</style></head>
<body><div class="myContainer" align="center">
    <div class="myImage"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(". myImage"). draggable({containment：$(". myContainer")，cursor："move"})用于实现拖动图像到页面中的任意位置。除此之外，draggable()方法还支持当移动到视区外时自动滚动文档，它通过设置 scroll 选项为 true 来启用自动滚动，当滚动触发时进行微调，滚动速度是通过 scrollSensitivity 和 scrollSpeed 选项设置的，例如 $(". myImage"). draggable({ scroll：true，scrollSensitivity：100 })。

需要说明的是，在使用 draggable()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA130. html。

096 以拖曳方式自动调整九宫格中的图像

此实例主要使用 jquery-ui. min. js 和 jquery-ui. min. css 中的 sortable()方法，从而实现以拖曳的方式将九宫格中的字母图像拖曳到新的位置。当在 Google Chrome 浏览器中显示该页面时，默认的

字母图像顺序如图 096-1 所示；使用鼠标拖动"A"到"C"和"I"之间，如图 096-2 所示，然后松开鼠标，则"A"就会被插入"C"和"I"之间。有关此实例的主要代码如下。



图 096-1



图 096-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {                               //以拖曳的方式将图像插入九宫格的指定位置
   $("#sortable").sortable();
   $("#sortable").disableSelection();    //禁用选择匹配的元素集合内的文本内容
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置盒子的基本样式*/
  .myContainer { width: 100%; height: 1500px; }
  /*设置无序列表的基本样式*/
  #sortable { list-style-type: none; margin: 0; padding: 0; width: 455px; }
  /*设置列表项的基本样式*/
  #sortable li { margin: 0; padding: 0; width: 150px; height: 150px;float: left; }
  /*设置列表项图像的基本样式*/
  li img { -webkit-transition: 0.3s ease-out;
         width:150px; height:150px;border:1px solid black; }
 </style></head>
<body><div class="myContainer" align="center">
 <ul id="sortable">
  <li><img src="img/A132A.jpg"></li><li><img src="img/A132B.jpg"></li>
  <li><img src="img/A132C.jpg"></li><li><img src="img/A132D.jpg"></li>
  <li><img src="img/A132E.jpg"></li><li><img src="img/A132F.jpg"></li>
  <li><img src="img/A132G.jpg"></li><li><img src="img/A132H.jpg"></li>
  <li><img src="img/A132I.jpg"></li></ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#sortable").sortable()用于实现以拖曳的方式将字母图像移动到九宫格的指定位置。sortable()方法是 jQuery UI 的可排序小部件(Sortable Widget)的方法，在任意的 DOM 元素上启用 sortable 功能，通过鼠标单击并拖曳元素到列表中的一个新的位置，其他条目就会自动调整。在默认情况下，sortable 的各个条目共享 draggable 属性。在此九宫格中，除了 sortable()方法以外，还需要控制 width：455px、width：150px、height：150px、float：left 等属性。在 CSS 中，float 属性用于定义元素在哪个方向浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。如果浮动非替换元素，则要指定一个明确的宽度，否则它们会尽可能地窄。假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。需要说明的是，在使用 sortable()方法实现拖动排序前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA132.html。

097 以固定长宽比例使用鼠标拖曳缩放图像

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 resizable()方法中传递参数 aspectRatio，从而实现拖放图像的边框线并且以固定的长宽比例改变图像的大小。当在 Google Chrome 浏览器中显示该页面时，图像显示为正方形，使用鼠标拖动图像的下边框线或右边框线，则图像在缩放的过程中长宽比始终保持为 2∶1，如图 097-1 所示。有关此实例的主要代码如下。

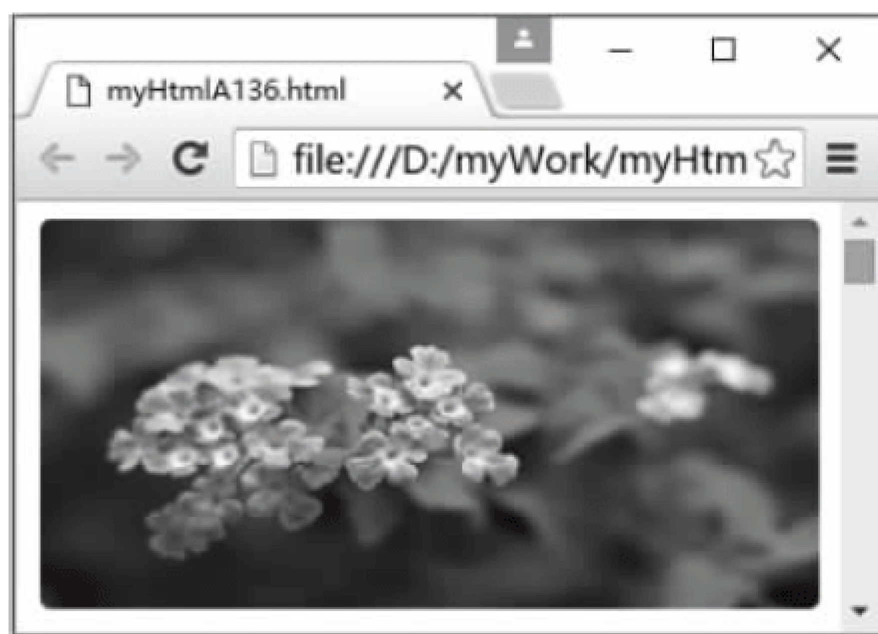


图　097-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {         //拖放图像的边框线并且以固定的长宽比例(2:1)改变图像的大小
   $(".myImage").resizable({aspectRatio: 2/1}); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  .myContainer { width: 100%; height: 1500px; }
  .myImage {width: 150px; height: 150px;background: url("img/A136.jpg");
            background-size: 100% 100%; border-radius: 5px;}
 </style></head>
<body><div class="myContainer" align="center">
    <div class="myImage"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(".myImage").resizable({aspectRatio: 2/1})用于拖放图像的边框线并且以固定的长宽比例(2:1)改变图像的大小。resizable()方法是可调整尺寸小部件(Resizable Widget)的方法，因此在使用 resizable()方法缩放图像之前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA136.html。

098　在拖曳缩放图像时限制拖曳的缩放范围

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 resizable()方法中传递参数 containment，从而实现在拖曳缩放图像时限制拖曳的缩放范围。当在 Google Chrome 浏览器中显示该页面时，图像显示为正方形，使用鼠标拖动图像的下边框线或右边框线即可改变图像的长宽尺寸，但是缩放范围始终限制在青色区域之内，如图 098-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {       //限制#myImage 的缩放范围只能在#myBox 中
   $("#myImage").resizable({containment: "#myBox" }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
 <style type="text/css">
  #myBox { width: 400px; height: 250px; background-color: cyan;}
```

```
   #myImage { background-position: top left; width: 100px; height: 100px;
            background: url("img/A137.jpg");background-size: 100% 100%; }
   #myImage, #myBox { padding: 0.5em; }
  </style></head>
 <body><div id="myBox" class="ui-widget-content" align="center">
  <div id="myImage" class="ui-state-active"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myImage").resizable({containment:"#myBox"})用于限制#myImage的缩放范围只能在#myBox中。resizable()方法主要用于使用鼠标改变元素的尺寸，该方法是可调整尺寸小部件(Resizable Widget)的方法，因此在使用resizable()方法缩放图像之前必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA137.html。



图　098-1

099　在一组图像中获取使用鼠标选择的图像

此实例主要使用jquery-ui.min.js和jquery-ui.min.css的selectable()方法，从而实现在一组图像中获取用户使用鼠标选择的一个或多个图像。当在Google Chrome浏览器中显示该页面时，如果使用鼠标选择第1、2、3、5几个图像，则这些图像的边框线将变成红色，并在下面显示选择结果，如图099-1所示。当然，所有图像都可以任意测试。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {       //获取多重选择结果
   $("#myItems").selectable({
    stop: function() {
     var myResult = $("#myResult").empty();
     $(".ui-selected", this).each(function () {
       var myIndex = $("#myItems li").index(this);
       if(myIndex >= 0)
         myResult.append(" 图" + (myIndex + 1 )); });} });});
 </script>
```

```
<link rel = "stylesheet" href = "css/jquery - ui.min.css">< style type = "text/css">
   * { margin: 0; padding: 0; }
  /* 设置盒子的基本样式 */
  .myContainer { margin: 20px 30px;}
  /* 设置无序列表的基本样式 */
  #myItems { list - style - type: none; margin: 0; padding: 0; width: 350px;}
  /* 设置列表项的基本样式 */
  #myItems li { margin: 5px; padding: 0px; width: 100px; height: 100px;float: left;}
  /* 设置列表项图像的基本样式 */
  li img { width: 100px; height: 100px;border: 0px solid black;}
  /* 设置已经选择的列表项基本样式 */
  #myItems .ui - selected { border: 1px solid red; }
</style></head>
<body><div class = "myContainer" align = "center"><ul id = "myItems">
  <li class = "ui - state - default">< img src = "img/A132A.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132B.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132C.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132D.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132E.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132F.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132G.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132H.jpg"></li>
  <li class = "ui - state - default">< img src = "img/A132I.jpg"></li></ul>
 <p><span>您已经选择了：</span> <span id = "myResult"></span></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，selectable()方法用于处理使用鼠标选择的单个元素或一组元素，参数中的 stop 表示在鼠标停止动作后执行这部分代码。需要说明的是，selectable()方法是可选择小部件(Selectable Widget)的方法，因此在使用 selectable()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA161.html。

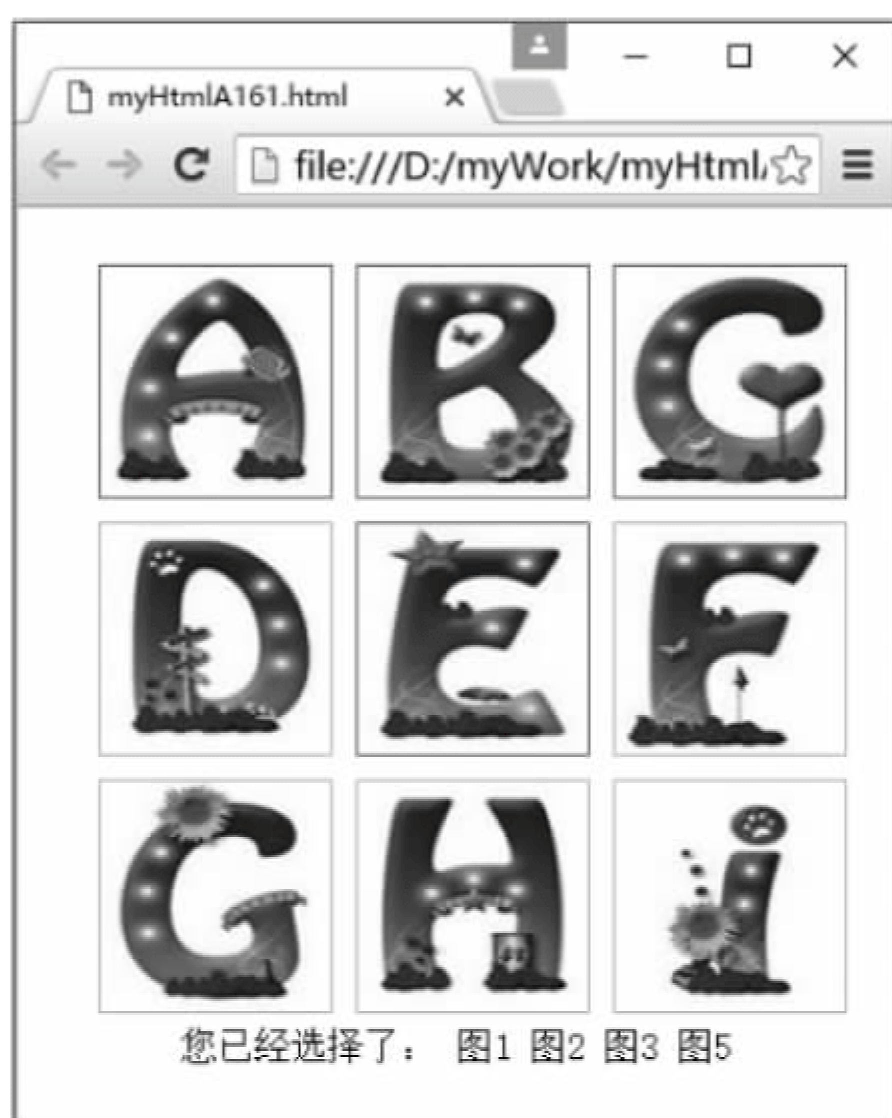


图 099-1

100 在移动鼠标时反色显示鼠标指针周围的图像

此实例主要在 mousemove 事件中设置反色滤镜的 CSS 样式，从而实现在鼠标移动时反色显示鼠标指针周围的图像。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针在图像上移动，则将对鼠标指针的当前位置附近的 50×50 区域的图像使用反色滤镜进行反色特效处理，效果如图 100-1 所示。有关此实例的主要代码如下。

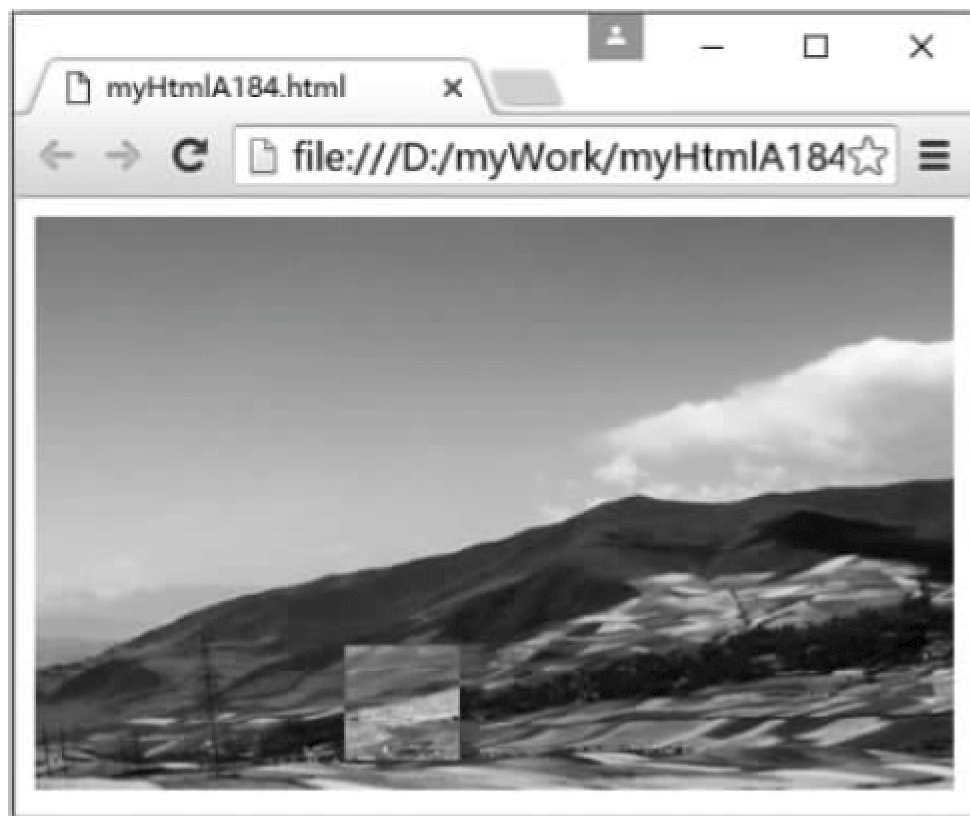


图 100-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   //在移动鼠标时反色滤镜跟随移动
   $("img").on("mousemove", function(e) {
    $(".myFilter").css("display","block");
    var x=e.offsetX+$(this).offset().left;
    var y=e.offsetY+$(this).offset().top;
    //根据鼠标指针的位置配置当前的反色滤镜
    $(".myFilter").css("top", y+"px");
    $(".myFilter").css("left", x+"px");
    $(".myFilter").css("background-position",
                   -e.offsetX+"px "+(-e.offsetY)+"px");
 });});
 </script>
 <style type="text/css">
  /*设置反色滤镜的基本样式*/
  .myFilter { position: absolute; width: 50px; height: 50px;display: none;
             -webkit-filter: invert(100%);background-image: url("img/B284.jpg");}
 </style></head>
<body><img src="img/B284.jpg"/><div class="myFilter"></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-filter: invert(100%)用于对元素进行完全的反色处理，即产生图像的底片效果。用户可以根据不同的需要调整反色参数的取值，范围为 0～100%，100%为完全反色。

此实例的源文件名是 myHtmlA184.html。

101 在移动鼠标时模糊显示鼠标指针周围的图像

此实例主要在 mousemove 事件中设置模糊滤镜的 CSS 样式，从而实现在鼠标移动时清晰地显示鼠标指针范围内的图像、模糊显示鼠标指针周围的图像。当在 Google Chrome 浏览器中显示该页面时，整幅图像较为模糊，如果鼠标指针在图像上移动，则将对鼠标指针的当前位置附近的 50×50 区域的图像使用模糊度为 0 的滤镜进行特效清晰处理，效果如图 101-1 所示。有关此实例的主要代码如下。

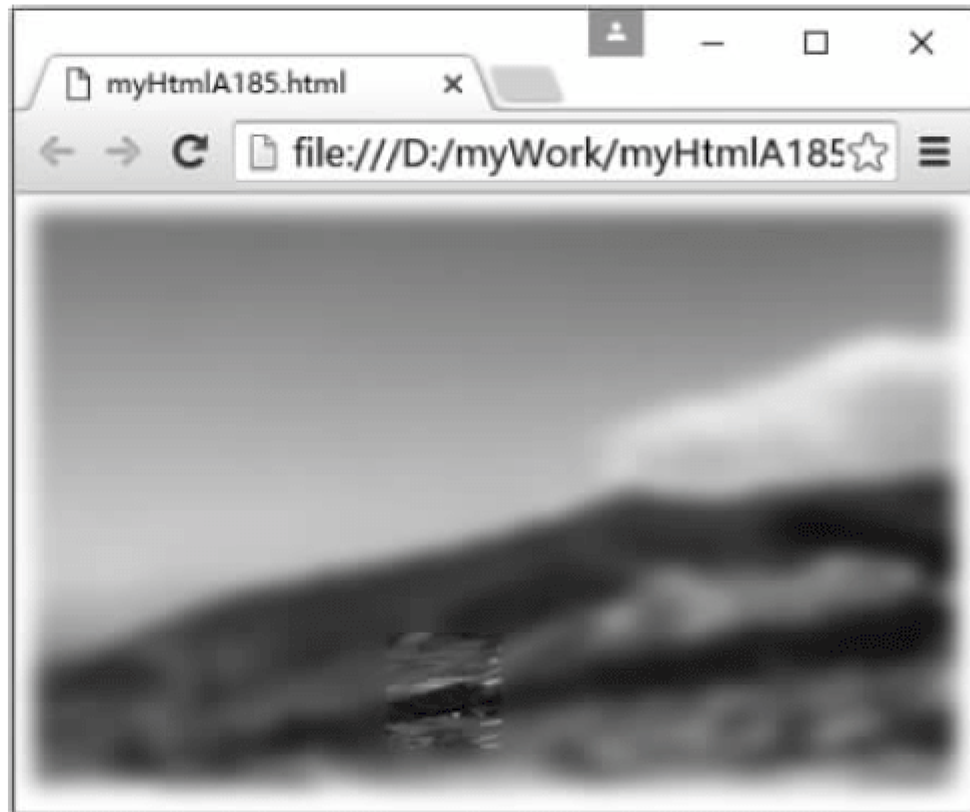


图 101-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("img").on("mousemove", function(e) {      //在移动鼠标时模糊滤镜跟随移动
     $(".myFilter").css("display", "block");
     var x = e.offsetX + $(this).offset().left;
     var y = e.offsetY + $(this).offset().top;
     //根据鼠标指针的位置配置当前的模糊滤镜
     $(".myFilter").css("top", y + "px");
     $(".myFilter").css("left", x + "px");
     $(".myFilter").css("background-position",
                          -e.offsetX + "px " + (-e.offsetY) + "px"); });});
 </script>
 <style type="text/css">
  /*设置模糊度为 0 的模糊滤镜的基本样式*/
  .myFilter { position: absolute; width: 50px; height: 50px; display: none;
           background-image: url("img/B284.jpg"); -webkit-filter: blur(0px); }
  /*以模糊滤镜处理图像*/
  img { display: block; -webkit-filter: blur(5px);}
 </style></head>
<body><img src="img/B284.jpg"/><div class="myFilter"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-filter：blur(5px)用于对元素进行模糊半径为 5px 的模糊特效处理，blur 的参数值越大，图像越模糊；参数值越小，图像越清晰，注意不能为负数，0px 即是原始图像。

此实例的源文件名是 myHtmlA185.html。

102　使用自定义方法对图像导圆角

此实例主要在 CanvasRenderingContext2D 对象上添加自定义方法 roundRect()，从而实现对画布上的图像导圆角。当在 Google Chrome 浏览器中显示该页面时，单击“直角显示图像”按钮，图像将以默认的直角效果显示；单击“圆角显示图像”按钮，图像在导圆角之后的显示效果如图 102-1 所示。有关此实例的主要代码如下。



图　102-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
  //自定义对图像进行圆角的方法
  CanvasRenderingContext2D.prototype.roundRect = function(x, y, w, h, r) {
   var min_size = Math.min(w, h);
   if (r > min_size/2) r = min_size/2;
    this.beginPath();
    this.moveTo(x + r, y);
    this.arcTo(x + w, y, x + w, y + h, r);
    this.arcTo(x + w, y + h, x, y + h, r);
    this.arcTo(x, y + h, x, y, r);
    this.arcTo(x, y, x + w, y, r);
    this.closePath();
    return this; }
  $(function(){
   var myContext = $("#myCanvas")[0].getContext("2d");
   var image = new Image();
   image.src = "img/A128.jpg";
   $("#myBtnRadius").click(function(){       //圆角显示图像
    myContext.clearRect(0,0,400,250);
    image.onload = function() {
     var pattern = myContext.createPattern(this, "no - repeat");
     myContext.roundRect(0, 0, this.width, this.height, 20 || 0);
     myContext.fillStyle = pattern;
     myContext.fill(); };
    image.onload(); });
```

```
        $("#myBtnDefault").click(function(){        //直角显示图像
         myContext.clearRect(0,0,400,250);
         myContext.drawImage(image,0,0,400,250); });});
    </script>
    <style type="text/css">
     button{ width:195px; }
    </style></head>
  <body><div align="center">
    <p><button id="myBtnRadius">圆角显示图像</button>
       <button id="myBtnDefault">直角显示图像</button></p>
    <canvas id="myCanvas" width="400" height="250"></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，CanvasRenderingContext2D.prototype.roundRect 是自定义的圆角方法。在 HTML5 中，CanvasRenderingContext2D 对象提供了一组用来在画布上绘制的图形方法，例如可以使用该对象的 strokeRect()和 fillRect()绘制矩形的边框和填充矩形，也可以使用 clearRect()清除矩形所定义的区域，此外，还可以使用该对象的 prototype 属性添加自定义方法和属性。通过 prototype 属性添加自定义的方法与普通的自定义方法基本一致，只需要在 prototype 属性后面指定一个自定义方法名称和对应的执行代码即可。

此实例的源文件名是 myHtmlA128.html。

103　将普通图像的 4 个直角改变为 4 个圆角

此实例主要在 CSS 样式中设置 border-radius 属性，从而实现将普通图像的 4 个直角改变为 4 个圆角。当在 Google Chrome 浏览器中显示该页面时，单击"对图像进行圆角"按钮，则图像经过圆角处理之后的效果如图 103-1 所示。有关此实例的主要代码如下。



图　103-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnRadius").click(function() {    //对图像进行圆角
    $("img").css("border-top-left-radius","50px");
    $("img").css("border-top-right-radius","50px");
    $("img").css("border-bottom-right-radius","50px");
    $("img").css("border-bottom-left-radius","50px"); });});
 </script>
 <style type="text/css">
  img { margin: 20px;}
 </style></head>
<body><input type="button" value="对图像进行圆角" id="myBtnRadius" style="margin-left: 20px;
width:390px;"/>
<div><img src="img/B002.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("border-top-left-radius","50px")用于设置图像左上角的圆角半径为50px，其余以此类推。border-radius属性是一个简写属性，用于设置元素的4个border-*-radius属性，例如border-radius:60px表示设置图像左上角、右上角、右下角和左下角的圆角半径都为60px。如果省略bottom-left，则与top-right相同；如果省略bottom-right，则与top-left相同；如果省略top-right，则与top-left相同。

此实例的源文件名是myHtmlB002.html。

104　对圆角图像添加可定制模糊效果的阴影

此实例主要在CSS样式中设置box-shadow属性，从而实现在圆角图像上定制具有模糊效果的阴影。当在Google Chrome浏览器中显示该页面时，单击"为图像添加阴影"按钮，圆角图像添加阴影之后的效果如图104-1所示。有关此实例的主要代码如下。



图　104-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
```

```
    $ ("#myBtnShadow").click(function() {        //为图像添加阴影
     $ ("img").css("box-shadow","10px 10px 5px gray"); });});
  </script></head>
 <body><input type="button" value="为图像添加阴影" id="myBtnShadow" style="margin-left: 10px;
 width:390px;"/>
 <div><img src="img/B004.jpg" style="margin: 10px;border-radius:20px;"/></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$ ("img").css("box-shadow","10px 10px 5px gray")用于设置圆角图像的阴影为灰色，它与 $ ("img").css("box-shadow","10px 10px 5px #888888")实现的功能类似。box-shadow 的语法说明如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

box-shadow 属性最多有 6 个参数设置，说明如下。

（1）阴影类型：此参数是一个可选值，如果不设置，其默认的投影方式是外阴影；如果取其唯一值“inset”，就是将外阴影变成内阴影，也就是说设置阴影类型为“inset”时其投影就是内阴影。

（2）x-offset：此参数是指阴影的水平偏移量，其值可以是正、负值，如果为正值，则阴影在对象的右边，反之其为负值时阴影在对象的左边。

（3）y-offset：此参数是指阴影的垂直偏移量，其值也可以是正、负值，如果为正值，阴影在对象的底部，反之其值为负值时阴影在对象的顶部。

（4）阴影模糊半径：此参数可选，但其值只能是正值，如果其值为 0，表示阴影不具有模糊效果，其值越大阴影的边缘越模糊。

（5）阴影扩展半径：此参数可选，其值可以是正、负值，如果为正值，则整个阴影都延展扩大，反之为负值，则缩小。

（6）阴影颜色：此参数可选，如果不设定任何颜色，浏览器会取默认色，但各浏览器的默认颜色不一样，建议用户不要省略此参数。

此实例的源文件名是 myHtmlB004.html。

105　在圆角图像外围添加扩散型的阴影

此实例主要在 CSS 样式中将 box-shadow 属性的 x-offset 和 y-offset 设置为 0，仅设置模糊半径 blur-radius 和扩散颜色 color，从而实现在圆角图像的外围添加扩散型的模糊阴影。当在 Google Chrome 浏览器中显示该页面时，单击“在图像的外围添加扩散型的阴影”按钮，将在圆角图像的外围添加绿色的模糊阴影，效果如图 105-1 所示。有关此实例的主要代码如下。

```
 <!doctype html><html><head><meta charset="UTF-8">
  <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $ (document).ready(function() {
    $ ("#myBtnShadow").click(function() {        //在图像的外围添加扩散型的阴影
     $ ("img").css("box-shadow", "0px 0px 30px green");
   });});
  </script></head>
 <body><input type="button" value="在图像的外围添加扩散型的阴影" id="myBtnShadow" style="margin
 -left: 10px;width: 390px;"/><div><img src="img/B004.jpg" style="margin: 10px;border-radius:
 20px;"/></div></body></html>
```



图 105-1

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("box-shadow","0px 0px 30px green")用于设置圆角图像的扩散颜色为绿色、模糊半径为 30px。box-shadow 的语法格式如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

在设置 box-shadow 属性值时，如果 x-offset 和 y-offset 的值为 0，则将在盒子的周围绘制阴影，即该实例所示的效果。用户也可以将阴影离开盒子的横向(偏移量)距离 x-offset 或阴影离开盒子的纵向距离 y-offset 设定为负数，如果将阴影离开盒子的横向距离 x-offset 设定为负数，则向左绘制阴影；如果将阴影离开盒子的纵向距离 y-offset 设定为负数，则向上绘制阴影。

此实例的源文件名是 myHtmlB044.html。

106　在圆角图像四周添加扩散的内置阴影

此实例主要在 CSS 样式中设置 box-shadow 属性的 inset 和模糊半径 blur-radius、阴影颜色 color，从而实现在圆角图像的内部添加扩散型的模糊阴影。当在 Google Chrome 浏览器中显示该页面时，单击“在图像的四周添加扩散型的内置阴影”按钮，则将在圆角图像的四周添加向内扩散的绿色的模糊阴影，效果如图 106-1 所示。有关此实例的主要代码如下。



图 106-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnShadow").click(function() {      //在图像的四周添加扩散型的内置阴影
   $("div").css("box-shadow", "0px 0px 150px rgba(45,255,98,0.8) inset");});});
 </script>
 <style>
  div {width: 390px; height: 250px; margin: 10px; border-radius: 20px;
       background-image: url(img/B004.jpg);background-size: 100% 100%;}
 </style></head>
<body><input type="button" value="在图像的四周添加扩散型的内置阴影" id="myBtnShadow" style=
"margin-left: 10px;width:390px;"/><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("box-shadow", "0px 0px 150px rgba(45,255,98,0.8) inset")用于设置圆角图像的扩散颜色为绿色、模糊半径为150px，并且是内置阴影，因为目标是图像，所以颜色应该设置一定的透明度。box-shadow 的语法格式如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

box-shadow 默认是外置阴影，因此可以省略 inset；当需要设置内置阴影时应该明确指定 inset。inset 可以在后面（如该实例所示），也可以在前面，例如“$("div").css("box-shadow", "inset 0px 0px 150px rgba(45,255,98,0.8)")”，测试表明均正确。

此实例的源文件名是 myHtmlB046.html。

107 在图像的下端添加阴影凸出显示图像

此实例主要在 CSS 样式中设置 box-shadow 属性的纵向偏移量 y-offset、模糊半径 blur-radius 和阴影颜色 color，从而实现在圆角图像的下端添加阴影以凸出样式显示图像。当在 Google Chrome 浏览器中显示该页面时，单击“在图像的下端添加阴影”按钮，将在圆角图像的下端产生凸出阴影，效果如图 107-1 所示。有关此实例的主要代码如下。



图 107-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
```

```
   $ (document).ready(function() {
    //在图像的下端添加阴影凸出显示图像
    $ ("#myBtnShadow").click(function() {
    $ ("img").css("box-shadow", "0 15px 10px rgba(13,30,4,0.8) ");});});
  </script></head>
<body><input type="button" value="在图像的下端添加阴影" id="myBtnShadow" style="margin-left:
10px;width:390px;"/><div><img src="img/B004.jpg" style="margin: 10px;border-radius:20px;"/></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$ ("img").css("box-shadow", "0 15px 10px rgba(13,30,4,0.8) ")用于在圆角图像的下端设置凸出阴影。box-shadow 的语法格式如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

在此实例中，15px 表示 y-offset 的偏移量，10px 表示模糊半径 blur-radius，rgba(13,30,4,0.8)表示阴影颜色 color。

此实例的源文件名是 myHtmlB047.html。

108 在圆角图像四周添加扩展的外置阴影

此实例主要在 CSS 样式中设置 box-shadow 属性的模糊半径 blur-radius、扩展半径 spread-radius 和阴影颜色 color，从而实现在圆角图像的外部添加扩展的模糊阴影。当在 Google Chrome 浏览器中显示该页面时，单击"在图像的四周添加阴影"按钮，将在圆角图像的四周添加淡淡的模糊阴影，如图 108-1 所示；单击"在图像的四周扩展阴影"按钮，将在圆角图像的四周添加厚重的模糊阴影，如图 108-2 所示。有关此实例的主要代码如下。



图 108-1



图 108-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $ (document).ready(function() {
```

```
    $("#myBtnShadow").click(function() {      //在图像的四周添加阴影
     $("img").css("box-shadow", "0 0 10px rgba(13,30,4,0.8)");
    });
    $("#myBtnExpand").click(function() {      //在图像的四周扩展阴影
     $("img").css("box-shadow", "0 0 10px 15px rgba(13,30,4,0.8)"); });});
 </script></head>
<body><input type="button" value="在图像的四周添加阴影" id="myBtnShadow" style=
"margin-left: 10px;width:190px;"/>
<input type="button" value="在图像的四周扩展阴影" id="myBtnExpand" style="margin-left: 10px;
width:190px;"/><div>
<img src="img/B004.jpg" style="margin: 20px;border-radius:20px;"/> </div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("box-shadow", "0 0 10px 15px rgba(13,30,4,0.8)")用于设置圆角图像的外围扩展阴影。box-shadow 的语法格式如下。

```
box-shadow: inset x-offset y-offset blur-radius spread-radius color
```

在此实例中，10px 表示模糊半径 blur-radius，15px 表示扩展半径 spread-radius，rgba(13,30,4,0.8)表示阴影颜色 color。

此实例的源文件名是 myHtmlB048.html。

109 通过对图像进行圆角实现裁剪椭圆图像

此实例主要在 CSS 样式中以百分比的形式设置 border-radius 属性，从而把任意图像裁剪为椭圆。当在 Google Chrome 浏览器中显示该页面时，单击“按照 10%裁剪圆角”按钮，图像按照 10%的比例裁剪圆角之后的效果如图 109-1 所示；单击“按照 50%裁剪圆角”按钮，图像按照 50%的比例裁剪圆角之后将成为一个标准的椭圆，效果如图 109-2 所示。有关此实例的主要代码如下。



图 109-1



图 109-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
```

```
    $("#myBtnRadius10").click(function() {      //按照 10% 裁剪圆角
     $("img").css("border-radius","10%");
    });
    $("#myBtnRadius25").click(function() {      //按照 25% 裁剪圆角
     $("img").css("border-radius","25%");
    });
    $("#myBtnRadius50").click(function() {      //按照 50% 裁剪圆角
     $("img").css("border-radius","50%"); });});
 </script></head>
<body><input type="button" value="按照 10% 裁剪圆角" id="myBtnRadius10" style="margin-left:
10px;width:120px;"/>
<input type="button" value="按照 25% 裁剪圆角" id="myBtnRadius25" style="margin-left: 10px;
width:120px;"/>
<input type="button" value="按照 50% 裁剪圆角" id="myBtnRadius50" style="margin-left: 10px;
width:120px;"/>
<div><img src="img/B003.jpg" style="margin: 10px;"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("border-radius","50%")表示按照图像的长度和宽度的50%作为4个角的圆角半径进行裁剪，而不论图像的长度和宽度的具体值是多少。这种情况特别适用于在图像未知的情况下进行圆角，例如允许用户对上传的图像进行圆角特效处理。

此实例的源文件名是 myHtmlB003.html。

110　设置 border-image 属性实现重复边框图案

此实例主要在 CSS 样式中设置 border-image 属性，从而实现使元素的边框线呈现不同的图案。当在 Google Chrome 浏览器中显示该页面时，单击"左右重复风格"按钮，则 div 元素的边框图案的显示风格如图 110-1 所示；单击"四边重复风格"按钮，则 div 元素的边框图案的显示风格如图 110-2 所示；单击"四边拉伸风格"按钮，则 div 元素的边框图案的显示风格如图 110-3 所示；单击"四边平铺风格"按钮，则 div 元素的边框图案的显示风格如图 110-4 所示。需要说明的是，虽然在 4 幅图中显示了 4 种不同的边框风格，但 4 种风格采用的均是同一张边框图案，如图 110-5 所示。有关此实例的主要代码如下。



图　110-1



图　110-2



图　110-3



图　110-4



图　110-5

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $ (document).ready(function() {
    $ ("#myBtnRepeat").click(function() {        //左右重复风格
     $ ("#myBorder").css("border - image",
                  "url(img/myBorder.jpg) 27/17px stretch repeat"); });
    $ ("#myBtnFull").click(function() {          //四边重复风格
       $ ("#myBorder").css("border - image",
                     "url(img/myBorder.jpg) 27/17px repeat"); });
    $ ("#myBtnStretch").click(function() {       //四边拉伸风格
       $ ("#myBorder").css("border - image",
                     "url(img/myBorder.jpg) 27/17px stretch"); });
     $ ("#myBtnRound").click(function() {        //四边平铺风格
       $ ("#myBorder").css("border - image",
                     "url(img/myBorder.jpg) 27/7px round"); });});
 </script>
 <style type = "text/css">
   #myBorder { width: 390px; height: 185px; text - align: center;}
 </style></head>
<body><p><input type = "button" value = "左右重复风格" id = "myBtnRepeat"/>
   <input type = "button" value = "四边重复风格" id = "myBtnFull"/>
   <input type = "button" value = "四边拉伸风格" id = "myBtnStretch"/>
   <input type = "button" value = "四边平铺风格" id = "myBtnRound"/></p>
<div id = "myBorder">
    <h3><br>日色欲尽花含烟,月明如素愁不眠。<br>
        <br>赵瑟初停凤凰柱,蜀琴欲奏鸳鸯弦。<br>
        <br>此曲有意无人传,愿随春风寄燕然。<br></h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myBorder").css("border-image","url(img/myBorder.jpg) 27/17px stretch repeat")中的 url()为图片链接，然后设置元素的边距为 27、宽度为 17，上下边框拉伸(stretch)，左右边框重复(repeat)；$("#myBorder").css("border-image","url(img/myBorder.jpg) 27/17px repeat")中的 repeat 表示边框的 4 边均重复。border-image 的重复性有别于 background 的背景重复，差别较大。background 图片包含重复、不重复、水平重复、垂直重复。而对于 border-image，repeat 只是其中之一，其余两个分别是 round 和 stretch，其中 stretch 是默认值。

此实例的源文件名是 myHtmlB001.html。

111 旋转多个图像模拟照片的不规则排列

此实例主要通过在 CSS 样式中设置 box-shadow 属性和使用 rotate()方法实现按照一定的角度旋转图像并设置阴影，以散列的效果显示多幅图像。当在 Google Chrome 浏览器中显示该页面时，带阴影的散列图像效果如图 111-1 所示。有关此实例的主要代码如下。



图 111-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  body{margin:30px;background-color:white;}
  div.myFrame{          /*创建有阴影的盒子*/
          width:190px;padding:10px;border:1px solid #BFBFBF;
          background-color:white;box-shadow:2px 2px 3px gray;}
  div.rotate_left{      /*向左旋转 7°*/
          float:left; transform:rotate(7deg);}
  div.rotate_right{     /*向右旋转 8°*/
          float:left;transform:rotate(-8deg);}
 </style></head>
<body><div class="myFrame rotate_left">
```

```
<img src="img/B005A.jpg" width="190" height="213" />
 <p style="color:darkred">中国中央电视台,简称央视,英语 China Central Television,简称 CCTV,是中华人民共和国国家电视台。</p></div>
<div class="myFrame rotate_right"><img src="img/B005B.jpg" width="190" height="213" /><p style="color:darkred">中国国家大剧院,由法国建筑师保罗·安德鲁主持设计,设计方为法国巴黎机场公司,是亚洲最大的剧院综合体。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,box-shadow:2px 2px 3px gray 用于创建有灰色阴影的盒子;transform:rotate(7deg)用于使元素(向左)旋转 7°。rotate(7deg)方法只有一个参数——角度,单位 deg 为度的意思,正数为顺时针旋转,负数为逆时针旋转。

此实例的源文件名是 myHtmlB005.html。

112 通过扭曲和旋转实现纸张的曲线投影

此实例主要在 CSS 样式中设置 box-shadow 属性和 transform 属性,通过产生扭曲和旋转来实现纸张的曲线投影特效。当在 Google Chrome 浏览器中显示该页面时,两张简历的左下角和右下角出现的曲线投影特效如图 112-1 所示。有关此实例的主要代码如下。



图 112-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  .curved_box { display: inline-block; width: 200px;height: 248px;
                margin: 20px; background-color: #FFF; border: 1px solid #EEE;
                box-shadow: 0 1px 4px rgba(0, 0, 0, 0.27),
                :0 0 40px rgba(0, 0, 0, 0.06) inset;
                position: relative;border-radius: 1px;}
  .curved_box:before {transform: skew(-15deg) rotate(-6deg);left: 15px;}
  .curved_box:after{transform: skew(15deg) rotate(6deg);right: 15px;}
  .curved_box:before, .curved_box:after {width: 70%; height: 55%;
            content: ''; box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
            position: absolute; bottom: 10px; z-index: -1;}
  p{padding-left: 15px; padding-right: 10px; font-size: 14px;}
  h3{text-align: center;}
 </style></head>
```

```
<body><div class="curved_box"><h3>徐海东</h3><p>徐海东(1900年6月17日-1970年3月25日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div>
<div class="curved_box"><h3>徐海东</h3><p>徐海东(1900年6月17日-1970年3月25日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,box-shadow用于向元素添加一个或多个阴影;transform:skew(15deg) rotate(6deg)中的skew(15deg)表示元素绕坐标轴倾斜15°,rotate(6deg)表示元素沿顺时针方向旋转6°,负数表示反向旋转。

此实例的源文件名是myHtmlB050.html。

113 通过旋转和圆角创建异形风格的头像

此实例主要通过设置元素的border-radius、transition、transform等属性创建异形风格的头像。当在Google Chrome浏览器中显示该页面时,头像显示的状态如图113-1所示;如果将鼠标指针放在左上角的头像上,会出现如图113-2所示的异形效果;如果将鼠标指针放在其他头像上,也会出现不同的异形效果。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myAvatar {margin: 10px; display: inline-block; width: 168px; height: 168px;
             border: 4px solid #00C7E9;border-radius: 87% 91% 98% 100%;
             -webkit-transition: all .35s;overflow: hidden;}
  .myAvatar:hover {border-radius: 95% 70% 100% 80%;transform: rotate(-2deg);}
  /*通过选择器定制头像*/
  .myAvatar:nth-child(2n+1) {border-radius: 59% 52% 56% 59%;
                            transform: rotate(-6deg);}
  .myAvatar:nth-child(2n+1):hover {border-radius: 51% 67% 56% 64% ;
                                  transform: rotate(-4deg);}
  .myAvatar:nth-child(3n+2) {border-radius: 84% 94% 83% 72%;
                            transform: rotate(5deg);}
  .myAvatar:nth-child(3n+2):hover {border-radius: 69% 64% 53% 70%;
                                  transform: rotate(0deg);}
  .myAvatar:nth-child(5n+3) {border-radius: 73% 100% 82% 100%;
                            transform: rotate(7deg);}
  .myAvatar:nth-child(5n+3):hover {border-radius: 73%;
                                  transform: rotate(7deg);}
  .myAvatar:nth-child(7n+5) {border-radius: 93% 90% 85% 78%;
                            transform: rotate(-2deg);}
  .myAvatar:nth-child(7n+5):hover {border-radius: 53% 70% 85% 68%;
                                  transform: rotate(-2deg);}
  .myAvatar:nth-child(11n+7) {border-radius: 68% 68% 83% 53%;
                             transform: rotate(-2deg);}
  .myAvatar:nth-child(11n+7):hover {border-radius: 58% 98% 78% 83%;
                                   transform: rotate(3deg);}
 </style></head>
<body><div><div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
```

```
<div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div>
 <div class="myAvatar"><img src="img/a043.jpg"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-radius：59% 52% 56% 59%表示以百分比的形式设置头像的 border-top-left-radius、border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius 几个位置的圆角半径；transform：rotate(－6deg)表示反方向旋转6°；-webkit-transition：all .35s 表示头像的所有属性在0.35 秒内完成从某个值改变到另一个值。

此实例的源文件名是 myHtmlB130.html。



图 113-1



图 113-2

114　以水平或垂直翻转的方式显示图像

此实例主要通过使用 rotateY()和 rotateX()方法实现对元素进行水平翻转或垂直翻转。当在 Google Chrome 浏览器中显示该页面时将显示未翻转的图像；单击"水平翻转"按钮，则水平翻转图像后的效果如图 114-1 所示；单击"垂直翻转"按钮，则垂直翻转图像后的效果如图 114-2 所示。有关此实例的主要代码如下。



图　114-1



图　114-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script language="javascript">
  $(document).ready(function() {
   $("#myBtnFlipH").click(function() {      //水平翻转
    $("img").css("-webkit-transform", "rotateY(180deg)");
   });
   $("#myBtnFlipV").click(function() {      //垂直翻转
    $("img").css("-webkit-transform", "rotateX(180deg)");
  }); });
  </script>
  <style type="text/css">
   img { width: 405px; height: 250px; border-radius: 5px; }
   input { width: 195px; border-radius: 2px;
           padding: 3px; margin: 2px;}
 </style></head>
<body><div><input type="button" value="水平翻转" id="myBtnFlipH"/>
     <input type="button" value="垂直翻转" id="myBtnFlipV"/></div>
  <img src="img/B147.jpg">
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，rotateY(angle)方法用于使元素实现沿着 Y 轴的 3D 旋转，参数 angle 表示旋转角度，范围为 0°～360°，当 angle 为 180deg 时则水平翻转元素；rotateX(angle)方法用于使元素实现沿着 X 轴的 3D 旋转，参数 angle 表示旋转角度，范围为 0°～360°，当 angle 为 180deg 时垂直翻转元素。

此实例的源文件名是 myHtmlB147.html。

115　使用旋转等方法创建心形风格的图形

此实例主要通过设置元素的 border-radius、transform、transform-origin 等属性创建心形图形。当在 Google Chrome 浏览器中显示该页面时，创建的心形图形效果如图 115-1 所示。有关此实例的主要代码如下。

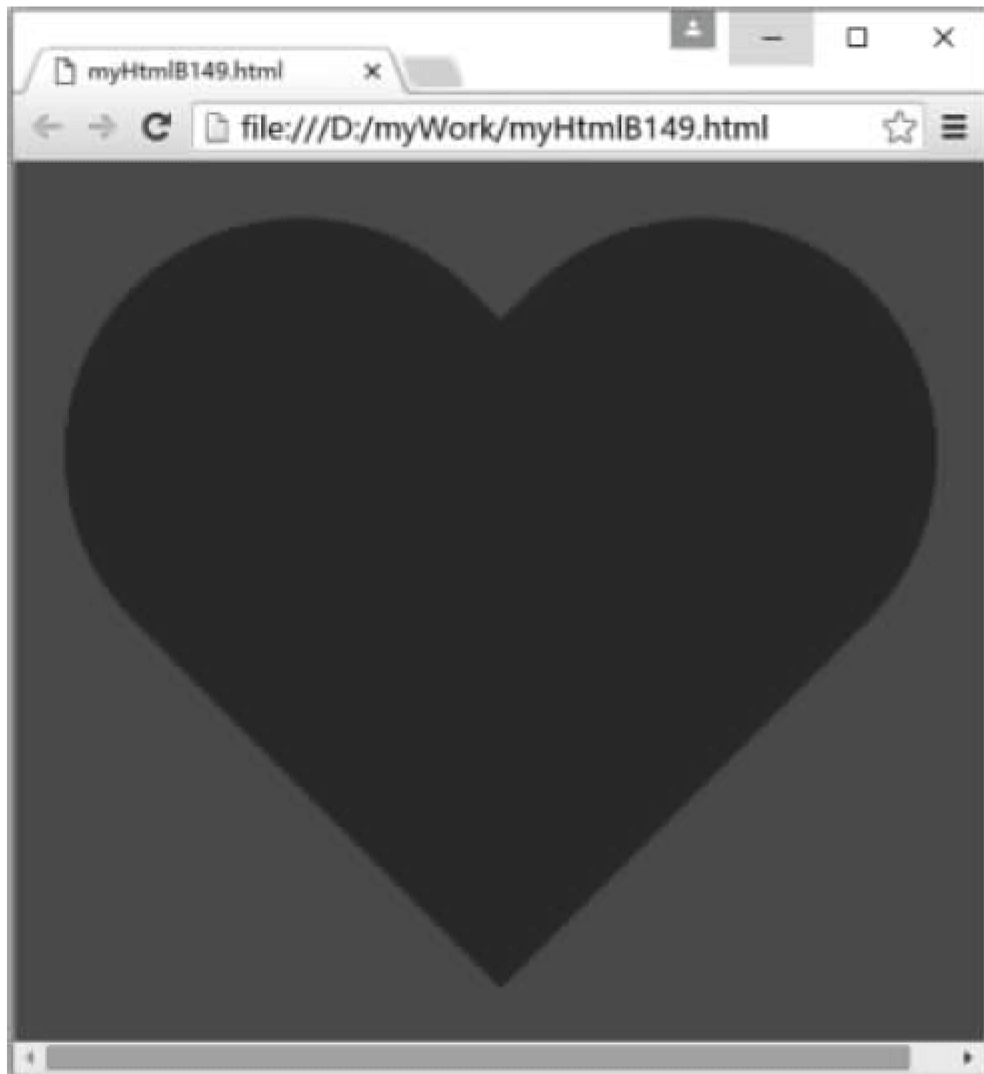


图　115-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   #heart { position: relative; }
   #heart:before, #heart:after { position: absolute; content: ""; left: 250px;
      top: 30px; width: 250px; height: 400px; background: darkred;
      border-radius: 250px 250px 0 0; -webkit-transform: rotate(-45deg);
      -webkit-transform-origin: 0 100%;}
   #heart:after {left: 0; -webkit-transform: rotate(45deg);
                  -webkit-transform-origin: 100% 100%;}
  body{ background: green;}
</style></head>
<body><div id="heart"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform 属性用于向元素应用 2D 或 3D 转换，即对元素进行旋转、缩放、移动或倾斜；-webkit-transform：rotate(45deg)表示将元素旋转 45°。transform-origin 属性用于改变被转换元素的位置，其语法格式如下。

```
transform-origin: x-axis y-axis z-axis;
```

其中，x-axis 用于定义视图被置于 X 轴的何处；y-axis 用于定义视图被置于 Y 轴的何处；z-axis 用于定义视图被置于 Z 轴的何处。

border-radius 属性是一个简写属性，用于设置元素的 border-top-left-radius、border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius(4 个角的半径值)，以产生圆角效果。

此实例的源文件名是 myHtmlB149. html。

116 使用旋转等方法创建无穷大符号

此实例主要通过设置元素的 border-radius、transform 等属性创建无穷大符号。当在 Google Chrome 浏览器中显示该页面时，创建的无穷大符号效果如图 116-1 所示。有关此实例的主要代码如下。



图 116-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #infinity { position: relative; width: 750px; height: 200px;
              -webkit-transform:scale(0.5,0.5); }
  #infinity:before, #infinity:after { content: ""; position: absolute;
    top: 0; left: 0; width: 240px;height:240px;
    border: 50px solid deepskyblue; border-radius: 200px 200px 0 200px;
    -webkit-transform: rotate(-45deg);}
  #infinity:after {left: auto; right: 0;border-radius: 200px 200px 200px 0;
                   -webkit-transform: rotate(45deg); }
  body { background: lightgray;}
 </style></head>
<body><div id="infinity"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform 属性用于向元素应用 2D 或 3D 转换，即对元素进行旋转、缩放、移动或倾斜；-webkit-transform: rotate(45deg)表示将元素旋转 45°；-webkit-transform:scale(0.5,0.5)表示将元素缩小一半；border-radius 属性是一个简写属性，用于设置元素的 border-top-left-radius、border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius(4 个角的半径值)，以产生圆角效果。

此实例的源文件名是 myHtmlB150.html。

117 根据图形裁剪绝对定位元素的局部区域

此实例主要通过设置绝对定位元素的 clip 属性实现根据指定的图形裁剪元素的部分区域。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素(div 块)，单击“剪切图像

的左右两端"按钮，该 div 块的左、右两端被裁剪，保留中间部分，如图 117-1 所示。有关此实例的主要代码如下。



图 117-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {      //剪切图像的左、右两端
    $("div").css("clip", "rect(auto 370px auto 60px)"); });});
 </script>
 <style type="text/css">
  div { width: 428px; height: 283px; border-radius: 5px; position: absolute;
        background-image: url(img/B088.jpg);text-align: center;}
  input {width: 420px;border-radius: 5px; margin: 5px;}
 </style></head>
<body><input type="button" value="剪切图像的左右两端" id="myBtnClip"/>
<div><h3>世界历史文化遗产</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("clip", "rect(auto 370px auto 60px)")用于指定裁剪或保留的区域。在 CSS3 中，clip 属性用于检索或设置对象的可视区域，区域外的部分是透明的，在使用此属性时必须将元素的 position 属性值设为 absolute 或者 fixed，这样此属性才可使用。clip 属性的语法格式如下。

```
clip: auto | <shape>
<shape>: rect(<number>|auto <number>|auto <number>|auto <number>|auto)
```

其中，各属性值的意义如下。

(1) auto：该值表示对象无剪切。

(2) rect(<number>|auto <number>|auto <number>|auto <number>|auto)：该值表示依据上-右-下-左的顺序提供自对象左上角为(0,0)坐标计算的 4 个偏移数值，其中任一数值都可用 auto 替换，即此边不剪切。例如 clip:rect(auto 50px 20px auto)表示上边不剪切，右边从左起第 50 个像素开始剪切直到最右边，下边从上起第 20 个像素开始剪切直到最底部，左边不剪切。

此实例的源文件名是 myHtmlB088.html。

118　将绝对定位元素的区域裁剪成六边形

此实例主要通过设置绝对定位元素的 clip-path 属性把元素的中心部分裁剪成一个六边形。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素(div 块),单击“将图像裁剪成一个正六边形”按钮,则该 div 块的周围被裁剪,中间部分保留成一个六边形,如图 118-1 所示。有关此实例的主要代码如下。



图　118-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnClip").click(function() {        //将图像裁剪成一个正六边形
     $("div").css("-webkit-clip-path",
       "polygon(0% 50%, 25% 0%, 75% 0%, 100% 50%, 75% 100%, 25% 100%)"); });});
 </script>
 <style type="text/css">
  div {width: 428px; height: 283px;border-radius: 5px;position: absolute;
       background-image: url(img/B088.jpg); text-align: center;
/* -webkit-clip-path: polygon(0% 50%, 25% 0%, 75% 0%, 100% 50%, 75% 100%, 25% 100%); */
}
  input{width:420px;border-radius: 5px; margin: 5px;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个正六边形" id="myBtnClip"/>
<div><h3>世界历史文化遗产</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("div").css("-webkit-clip-path","polygon(0% 50%, 25% 0%, 75% 0%, 100% 50%, 75% 100%, 25% 100%)")用于指定裁剪或保留的区域是一个六边形。在 CSS3 中,clip-path 属性是一个通过屏蔽和裁剪来隐藏元素的一部分的工具。尽管 clip-path 属性目前并没有被大多数浏览器支持,但在谷歌浏览器中它仍然是一个实现时尚效果的小工具。clip-path 简单的工作原理是提供一系列的 x 和 y 值来创建路径。当使用这些值创建一条完整路径时会把图像按照路径内部的尺寸进行裁剪。使用 clip-path 可以创建圆形、椭圆、多边

形等不同的形状，用户的创造力是唯一的限制。从上面的实例可以看出，每一对坐标描述的即是多边形每一个点的位置。

此实例的源文件名是 myHtmlB089.html。

119 将绝对定位元素的区域裁剪成五角星

此实例主要通过设置绝对定位元素的 clip-path 属性把元素的中心部分裁剪成一个五角星。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素（div 块），单击“将图像裁剪成一个正五角星”按钮，该 div 块的周围被裁剪，中间部分保留成一个五角星，如图 119-1 所示。有关此实例的主要代码如下。



图 119-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {       //将图像裁剪成一个正五角星
     $("div").css("-webkit-clip-path", "polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82%
100%, 50% 75%, 18% 100%, 31% 59%, 0 38%, 37% 38%)"); });});
 </script>
 <style type="text/css">
  div {width: 330px;height: 330px; border-radius: 5px;position: absolute;
       background-image: url(img/B090.jpg);text-align: center;}
  input{width:325px;border-radius: 5px;margin: 2px;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个正五角星" id="myBtnClip"/>
<div><h3>世界建筑精品</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("-webkit-clip-path", "polygon(50% 0%, 63% 38%, 100% 38%, 69% 59%, 82% 100%, 50% 75%, 18% 100%, 31% 59%, 0 38%, 37% 38%)")用于指定裁剪或保留的区域是一个五角星，每一对坐标描述的即是五角星的 10 个点的位置。

此实例的源文件名是 myHtmlB090.html。

120 将绝对定位元素的区域裁剪成椭圆

此实例主要使用 ellipse()方法设置绝对定位元素的 clip-path 属性，从而把元素的中心部分裁剪成一个椭圆。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素(div 块)，单击“将图像裁剪成一个椭圆”按钮，则该 div 块的周围被裁剪，中间部分保留成一个椭圆，如图 120-1 所示。有关此实例的主要代码如下。



图 120-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {      //将图像裁剪成一个椭圆
    $("div").css("-webkit-clip-path", "ellipse(49% 48% at 50% 50%)");
  });});
 </script>
 <style type="text/css">
  div { width: 413px; height: 287px;border-radius: 5px; position: absolute;
        background-image: url(img/B091.jpg); text-align: center;}
  input{ width:410px;border-radius: 5px;margin: 2px; }
 </style></head>
<body><input type="button" value="将图像裁剪成一个椭圆" id="myBtnClip"/>
<div><h3>世界建筑精品</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("-webkit-clip-path", "ellipse(49% 48% at 50% 50%)")用于指定裁剪或保留的区域是一个椭圆，其中 ellipse()是绘制椭圆的方法，49%表示椭圆的 X 轴半径是 $("div")宽度的 49%，48%表示椭圆的 Y 轴半径是 $("div")高度的 48%，at 关键字后面的 50% 50%用于指定椭圆中心的 x 坐标和 y 坐标占宽度和高度的比例均是 50%。

此实例的源文件名是 myHtmlB091.html。

121 将绝对定位元素的区域裁剪成圆形

此实例主要使用 circle()方法设置绝对定位元素的 clip-path 属性，从而把元素的中心部分裁剪成一个圆形。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素(div 块)，单击"将图像裁剪成一个圆形"按钮，则该 div 块的周围被裁剪，中间部分保留成一个圆形，如图 121-1 所示。有关此实例的主要代码如下。



图 121-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {       //将图像裁剪成一个圆形
    $("div").css("-webkit-clip-path", "circle(45% at 50% 50%)");
  });});
 </script>
 <style type="text/css">
  div { width: 385px; height: 356px;border-radius: 5px;position: absolute;
        background-image: url(img/B092.jpg);text-align: center; }
  input{ width:385px;border-radius: 5px; margin: 2px;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个圆形" id="myBtnClip"/>
<div><h3>世界绝美风光</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("-webkit-clip-path", "circle(45% at 50% 50%)")用于指定裁剪或保留的区域是一个圆，其中 circle()是绘制圆的方法，45%表示圆的半径是$("div")宽度的 45%，at 关键字后面的 50% 50%用于指定圆中心的 x 坐标和 y 坐标占宽度和高度的比例均是 50%。

此实例的源文件名是 myHtmlB092.html。

122　将绝对定位元素的区域裁剪成三角形

此实例主要使用 polygon()方法设置绝对定位元素的 clip-path 属性，从而把元素的中心部分裁剪成一个三角形。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素（div 块），单击“将图像裁剪成一个三角形”按钮，则该 div 块的周围被裁剪，中间部分保留成一个三角形，如图 122-1 所示。有关此实例的主要代码如下。



图　122-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {       //将图像裁剪成一个三角形
    $("div").css("-webkit-clip-path", "polygon(0 99%, 50% 0, 98% 98%)");
  });});
 </script>
 <style type="text/css">
  div {width: 382px;height: 273px; border-radius: 5px;position: absolute;
       background-image: url(img/B093.jpg);text-align: center;}
  input{ width:382px;border-radius: 5px; margin: 2px;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个三角形" id="myBtnClip"/>
<div><h3>世界绝美风光</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("-webkit-clip-path", "polygon(0 99%, 50% 0, 98% 98%)")用于指定裁剪或保留的区域是一个三角形，其中 polygon()是绘制多边形的方法，0 99%表示三角形左下角顶点的 x 坐标是$("div")宽度的 0%、y 坐标是$("div")高度的 99%；50% 0 表示三角形上端顶点的 x 坐标是$("div")宽度的 50%、y 坐标是$("div")高度的 0%；98% 98%表示三角形右下角顶点的 x 坐标是$("div")宽度的 98%、y 坐标是$("div")高度的 98%。

此实例的源文件名是 myHtmlB093.html。

123　将绝对定位元素裁剪成内投影图形

此实例主要使用 inset()方法设置绝对定位元素的 clip-path 属性，从而将绝对定位元素的部分区域裁剪成内投影风格的图形。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对定位元素(div 块)，单击"将图像裁剪成一个内投影图形"按钮，该 div 块的周围被裁剪，指定部分保留成一个内投影风格的图形，如图 123-1 所示。有关此实例的主要代码如下。



图　123-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {       //将图像裁剪成一个内投影图形
   $("div").css("-webkit-clip-path", "inset(35% 0 15% 0 round 0 24% 0 26%)");});});
  </script>
  <style type="text/css">
   div { width: 382px; height: 273px;border-radius: 5px;position: absolute;
         background-image: url(img/B094.jpg); text-align: center; }
   input{width:382px;border-radius: 5px;margin: 2px;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个内投影图形" id="myBtnClip"/>
<div><h3>世界绝美风光</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("-webkit-clip-path", "inset(35% 0 15% 0 round 0 24% 0 26%)")用于指定裁剪或保留的区域是一个内投影图形，其中 inset(35% 0 15% 0 round 0 24% 0 26%)的值对应 inset(<top><right><bottom><left> round <top-radius><right-radius><bottom-radius><left-radius>)。

此实例的源文件名是 myHtmlB094.html。

124　将绝对定位元素裁剪成手柄式图形

此实例主要使用 polygon()方法设置绝对定位元素的 clip-path 属性，从而将绝对定位元素的部分区域裁剪成手柄风格的图形。当在 Google Chrome 浏览器中显示该页面时将显示带背景图像的绝对

定位元素(div 块),单击"将图像裁剪成一个手柄图形"按钮,则该 div 块的周围被裁剪,指定部分保留成一个手柄风格的图形,如图 124-1 所示。有关此实例的主要代码如下。



图 124-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnClip").click(function() {       //将图像裁剪成一个手柄图形
    $("div").css("-webkit-clip-path","polygon(0% 0%, 100% 0%, 100% 75%, 78% 75%, 90%
100%, 40% 75%, 0% 75%)");
  });});
 </script>
 <style type="text/css">
  div {width: 412px;height: 323px;border-radius: 5px;position: absolute;
       background-image: url(img/B095.jpg);text-align: center;}
  input{width:410px;border-radius: 5px;margin: 2px;}
  h3{color:yellow;}
 </style></head>
<body><input type="button" value="将图像裁剪成一个手柄图形" id="myBtnClip"/>
<div><h3>世界绝美风光</h3></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("div").css("-webkit-clip-path","polygon(0% 0%, 100% 0%, 100% 75%, 78% 75%, 90% 100%, 40% 75%, 0% 75%)")用于指定裁剪或保留的区域是一个手柄风格的图形,其中 polygon(0% 0%, 100% 0%, 100% 75%, 78% 75%, 90% 100%, 40% 75%, 0% 75%)以逗号分隔的每对值表示以图像的左上角为起点沿顺时针方向的每个拐点的坐标值,该坐标值的前一个 x 坐标是$("div")宽度的百分比值,后一个 y 坐标是$("div")高度的百分比值。

此实例的源文件名是 myHtmlB095.html。

125 根据指定的位置和大小截取图片内容

此实例主要设置子元素的 object-position 属性,从而实现根据指定的位置和大小截取图片的内容。当在 Google Chrome 浏览器中显示该页面时,如果在文本框中输入或选择一个数字,将在下面以

带圈的形式显示输入的数字，如图 125-1 所示。例如 2、9、9、6 这 4 个带圈数字实际上是 4 幅图像，它们都是根据指定的位置和大小从一幅图像中截取出来的。有关此实例的主要代码如下。



图　125-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("input").change(function() {
    var value = this.value.trim() * 1;
    if (!value || value < 1000 || value > 9999) { value = 1283; }
    this.value = value;
    this.value.split("").forEach(function (num, index) {
      $(".num")[index].className = "num num" + num; });});});
 </script>
 <style>
  .num { width: 40px;height: 40px; object-fit: none;object-position: 0 0;
        -webkit-transition: object-position .25s;}
  .num0 { }
  .num1 {object-position: 0 -40px;}
  .num2 {object-position: 0 -80px;}
  .num3 {object-position: 0 -120px;}
  .num4 {object-position: 0 -160px;}
  .num5 {object-position: 0 -200px;}
  .num6 {object-position: 0 -240px;}
  .num7 {object-position: 0 -280px;}
  .num8 {object-position: 0 -320px;}
  .num9 {object-position: 0 -360px;}
  .myContainer { margin: auto;width: -webkit-fit-content; }
  div { width: 300px; }
 </style></head>
<body><p><strong>支持显示的数字是(1000～9999): <input type = "number" value = "1283" min = "1000"
max = "9999"></strong></p>
<div><p class = "myContainer">
      <img src = "img/B139.png" class = "num num1">
      <img src = "img/B139.png" class = "num num2">
      <img src = "img/B139.png" class = "num num8">
      <img src = "img/B139.png" class = "num num3"></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<img src="img/B139.png" class="num num1">表示在 B139.png 图像中的“0 −40px”位置截取“width：40px;height：40px”大小的图像，即带圈数字 1。

此实例的源文件名是 myHtmlB139.html。

126 通过裁剪方式获取大图像的局部内容

此实例主要通过设置 clip 属性实现以裁剪的方式显示整个图像的部分内容。当在 Google Chrome 浏览器中显示该页面时，默认的五角星以灰色显示，如图 126-1 所示；将鼠标指针放在五角星上，则显示黄色的五角星，如图 126-2 所示。灰色的五角星和黄色的五角星实际上都在 B163.png 中，只是在显示时进行了裁剪。有关此实例的主要代码如下。



图 126-1



图 126-2

```
<!doctype html><html><head><meta charset = UTF - 8>
 <style type = "text/css">
   .clip_a { width: 20px;height: 20px; }
   /* 剪裁上半部分的五角星 */
   .clip_a img { border: 0;position: absolute; clip: rect(0 20px 20px 0);}
   /* 剪裁下半部分的五角星 */
   .clip_a:hover img { margin - top: - 20px;clip: rect(20px 20px 40px 0); }
 </style></head>
<body><span>请将鼠标放在五角星图片上试试: </span>
<a href = "javascript:" class = "clip_a"><img src = "img/B163.png"/></a></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，clip：rect(0 20px 20px 0)用于裁剪灰色的五角星。在 CSS 中，clip 属性用于剪裁绝对定位元素，对于一个绝对定位元素，在这个矩形内的内容才可见，超出了这个剪裁区域的内容会根据 overflow 属性的值来处理，剪裁区域可能比元素的内容区大，也可能比内容区小。clip 属性的语法格式如下。

```
clip:shape|auto|inherit
```

其中，各属性值的意义如下。

(1) shape：该属性值设置元素的形状，例如 rect (top right bottom left)。

(2) auto：默认值，表示不应用任何剪裁。

(3) inherit：该属性值规定应该从父元素继承 clip 属性的值。

此实例的源文件名是 myHtmlB188.html。

127 使用遮罩实现以不规则形状裁剪图像

此实例主要通过设置-webkit-mask-box-image 属性实现根据不规则形状裁剪图像。当在 Google Chrome 浏览器中显示该页面时将显示未裁剪的图像；单击“裁剪图像”按钮，将按照如图 127-1 所示的形状裁剪图像，效果如图 127-2 所示。有关此实例的主要代码如下。

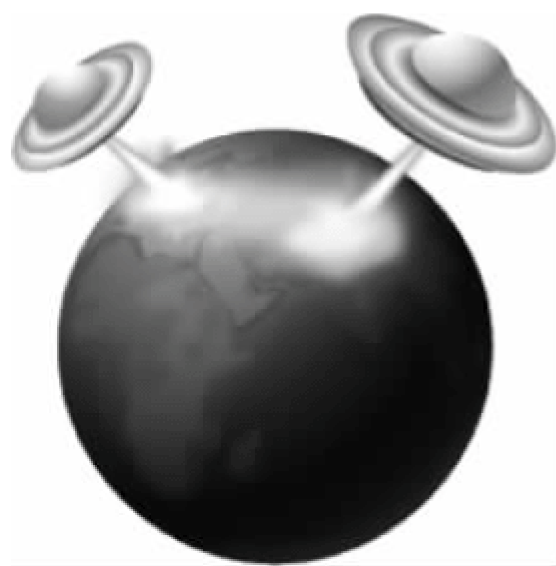

图 127-1



图 127-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#myInitial").click(function(){       //原始图像
    $(".myImage").css("-webkit-mask-box-image","initial");
   });
   $("#myMask").click(function(){          //裁剪图像
    $(".myImage").css("-webkit-mask-box-image","url(img/B302.png)");
  });});
 </script>
 <style type="text/css">
  .myImage{ margin-top:2px; overflow: hidden; width: 350px; height:256px;
            background-image: url(img/B114.jpg);background-size: 100% 100%; }
  button{ width:170px; }
 </style></head>
<body><div align="center">
  <button id="myInitial">原始图像</button><button id="myMask">裁剪图像</button>
<div class="myImage"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(".myImage").css("-webkit-mask-box-image","url(img/B302.png)")用于按照B302.png图像的形状裁剪当前$('img')图像。需要说明的是，-webkit-mask-box-image不是标准属性，在正式产品中使用时需要慎重考虑。-webkit-mask-box-image的语法格式如下。

```
-webkit-mask-box-image: <mask-box-image> [<top><right><bottom><left><x-repeat><y-repeat>]
```

其中，<mask-box-image>可以是<uri> | <gradient> | none，<top><right><bottom><left>可以是<length> | <percentage>，<x-repeat><y-repeat>可以是 repeat | stretch | round。

此实例的源文件名是myHtmlB339.html。

128 使用遮罩实现以 png 图像形状裁剪图像

此实例主要通过设置-webkit-mask-image属性实现使用遮罩技术根据png图像的形状裁剪图像。当在Google Chrome浏览器中显示该页面时将显示未裁剪的图像；单击“使用Google Glass眼

镜 PNG 图标裁剪图像特效一”按钮，则图像按照 png 图标的形状裁剪后的效果如图 128-1 所示；单击“使用 Google Glass 眼镜 PNG 图标裁剪图像特效二”按钮，图像按照 png 图标的形状裁剪后的效果如图 128-2 所示。有关此实例的主要代码如下。



图 128-1



图 128-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><style type="text/css">
  button { width: 512px; margin: 2px; }
  p { text-align: center; }
   * { margin: 0; }
 </style></head>
<body><p><button onclick="$('img').css('-webkit-mask-image', 'url(img/B207A.png)')">使用 Google
Glass 眼镜 PNG 图标裁剪图像特效一</button>
   <button onclick="$('img').css('-webkit-mask-image', 'url(img/B207B.png)')">使用 Google Glass
眼镜 PNG 图标裁剪图像特效二</button>
    <p><img src="img/B207.jpg"/></p></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$('img').css('-webkit-mask-image','url(img/B207A.png)')"用于实现当前 $('img')图像按照 B207A.png 图标的形状进行裁剪。在 CSS 中，遮罩提供了一种基于像素级别的可以控制元素透明度的能力，类似于 24 位 png 或 32 位 png 中的 alpha 透明通道的效果。图像是由 R、G、B 几个通道以及在每个像素上定义的颜色组成的。在它们之上还有第 4 个通道，即 alpha 通道，通过亮度定义每个像素上的透明度。白色意味着不透明，黑色意味着透明，介于黑、白之间的灰色表示半透明。在 html 元素中使用 css 遮罩时不需要给图片应用一个 alpha 通道，只需要设置元素的-webkit-mask-image 属性，例如-webkit-mask-image：url(mouse.png)，这样就能够从图片遮罩里读出图片的透明信息，然后应用到 html 元素上。

此实例的源文件名是 myHtmlB207.html。

129 以不同位置为原点放大或缩小图像

此实例主要通过设置元素的 transform-origin 属性实现以左上角(或中心位置)为原点对图像进行放大或缩小。在 CSS3 中,zoom 和 scale 这两个方法都是用于对元素进行缩放,但两者除了兼容性之外还有一些不同的地方,例如 zoom 缩放会将元素保持在左上角,而 scale 默认是中间位置,可以通过 transform-origin 来设置;另外两者执行的渲染顺序也不同,zoom 可能影响到盒子的计算。当在 Google Chrome 浏览器中显示该页面时,单击"以左上角为原点缩小图像"按钮,图像缩小至 80%后的效果如图 129-1 所示;单击"以图像中心为原点缩小图像"按钮,图像缩小至 80%后的效果如图 129-2 所示。有关此实例的主要代码如下。



图 129-1



图 129-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnLeft").click(function() {          //以左上角为原点缩小图像
    $("img").css("transform-origin", "top left");
    $("img").css("transform", "scale(0.8)");
   });
   $("#myBtnCenter").click(function() {        //以图像中心为原点缩小图像
    $("img").css("transform-origin", "");
    $("img").css("transform", "scale(0.8)"); });});
 </script>
 <style type="text/css">
  div{width:400px; height:250px;border: 1px solid gray; border-radius: 5px; }
  img{ border-radius: 5px; }
  input { width: 195px;}
 </style></head>
<body><p><input type="button" value="以左上角为原点缩小图像" id="myBtnLeft"/>
   <input type="button" value="以图像中心为原点缩小图像" id="myBtnCenter"/></p>
<div><img src="img/B128.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,transform-origin 属性用于设置 transform 元素的基点位置,transform-origin 属性的语法格式如下。

```
transform - origin: x - axis y - axis z - axis;
```

其中，各属性值的说明如下。

(1) x-axis：用于定义视图被置于 X 轴的何处，可能的值为 left、center、right、length、%等。
(2) y-axis：用于定义视图被置于 Y 轴的何处，可能的值为 top、center、bottom、length、%。
(3) z-axis：用于定义视图被置于 Z 轴的何处，可能的值为 length。

此实例的源文件名是 myHtmlB128.html。

130 通过上下按动鼠标滚轮实现图像缩放

此实例主要在 img 元素的鼠标滚轮 mousewheel 事件中使用 scale()方法，从而实现上下按动鼠标滚轮放大或缩小图像的效果。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在图像上，向下滑动滚轮则缩小图像，如图 130-1 所示；向上滑动滚轮则放大图像，如图 130-2 所示。有关此实例的主要代码如下。

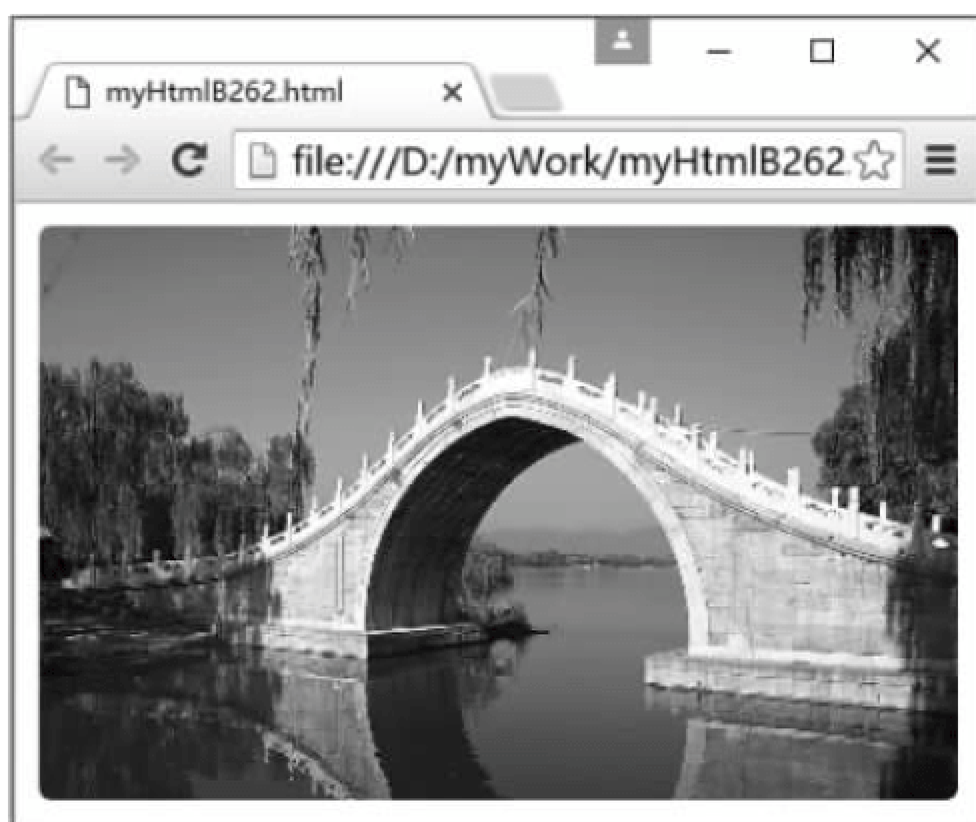


图 130-1



图 130-2

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
  $ (function() {
   var myWheel = 0, myFactor = 1;                    //myFactor 为缩放因子
   $ ('img').on('mousewheel', function() {           //为 img 元素添加鼠标滚轮监测事件
     myWheel = event.wheelDelta > 0 ? 1 : - 1;
     if (myWheel > 0) {                              //鼠标滚轮向上滚动
       myFactor += 0.1;
     } else if (myWheel < 0) {                       //鼠标滚轮向下滚动
       myFactor -= 0.1; }
     $ ('img').css(' - webkit - transform', 'scale(' + myFactor + ')'); });});
 </script>
 <style type = "text/css">
  div { width: 400px; height: 250px;
        /* 超出盒子范围自动裁剪 */
        overflow: hidden; margin: 10px auto;border - radius: 5px; }
  img { border - radius: 5px; }
 </style></head>
<body><div><img src = "img/B262.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$('img').css('-webkit-transform','scale('+myFactor+')')表示根据缩放因子 myFactor 放大或缩小图像，缩放因子 myFactor 是根据鼠标滚轮 mousewheel 的状态进行设置的。在 CSS3 中 scale()方法用于对元素进行缩放，该方法的语法格式如下。

```
scale(x,y)
```

其中，x 表示水平方向的缩放倍数；y 表示垂直方向的缩放倍数，y 是一个可选参数，如果没有设置此参数，则表示 x、y 两个方向的缩放倍数是一样的。

此实例的源文件名是 myHtmlB262.html。

131　对图像局部区域进行毛玻璃状的模糊

此实例主要设置子元素的 background 属性值为 inherit，从而实现对图片的局部区域进行毛玻璃状的模糊。当在 Google Chrome 浏览器中显示该页面时，拖动左上角的方块到图像的任意位置，例如人像的左脸，则该左脸以模糊状态显示，其他部分仍然以清晰状态显示，如图 131-1 所示。有关此实例的主要代码如下。

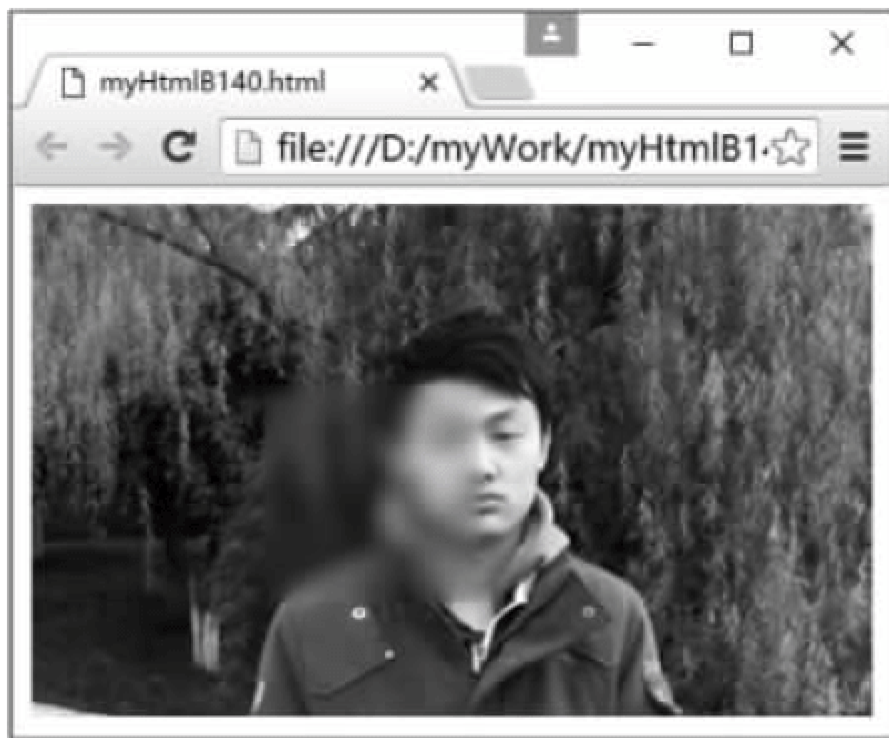


图　131-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(function() {
   var _move = false;                          //移动标记
   var _x, _y;                                 //鼠标指针离元素左上角的相对位置
   $(".drag").click(function() {
    //alert("click");                          //单击(松开后触发)
   }).mousedown(function(e) {
    _move = true;
    _x = e.pageX - parseInt($(".drag").css("left"));
    _y = e.pageY - parseInt($(".drag").css("top"));
   });
   $(document).mousemove(function(e) {
    if (_move) {
     var x = e.pageX - _x;                     //移动时根据鼠标指针的位置计算元素左上角的绝对位置
     var y = e.pageY - _y;
     $(".drag").css({ top: y, left: x });      //元素的新位置
```

```
        //动态偏移.drag元素背景图片的backgroundPosition位置,使其与背景图片重合
        var myX = ( - 1 * x); var myY = ( - 1 * y);
        $ (".drag").css("backgroundPosition", myX + "px " + myY + "px");
      } }).mouseup(function() {
        _move = false; });});
    </script>
    <style>
     .box { width: 385px; height: 235px;background: url(img/B004.jpg) no - repeat;
            position: relative; overflow: hidden;}
     .drag { width: 100px; height: 100px; background: inherit;
             - webkit - filter: blur(5px); cursor:  - webkit - grab;
             position: absolute;overflow: hidden;top: 1px;left: 1px;}
     .drag:active { cursor:  - webkit - grabbing; }
   </style></head>
  <body><div class = "box"><div id = "drag" class = "drag"></div></div>
  </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-filter: blur(5px)用于模糊子元素; background: inherit表示子元素的背景图像来自容器的背景图像; $(".drag").css("backgroundPosition",myX+ "px " +myY+ "px")表示在拖动子元素时从backgroundPosition指定的位置截取容器的背景图像,因此在拖动方块时方块显示的模糊图像始终是下面容器的对应位置的背景图像。

此实例的源文件名是myHtmlB140.html。

132 对图像和颜色以差值混合模式生成特效

此实例主要设置元素的mix-blend-mode属性为difference,从而实现以差值模式混合容器的背景颜色和图像。当在Google Chrome浏览器中显示该页面时,<img>元素显示的图像位于蓝色背景的容器<div>中;单击"以差值模式混合图像和颜色"按钮,则图像和蓝色经过差值模式混合后的效果如图132-1所示。有关此实例的主要代码如下。



图 132-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnDifference").click(function() {       //以差值模式混合图像和颜色
    $("img").css("mix-blend-mode", "difference");
  });});
 </script>
 <style type="text/css">
  .box { width: 400px; height: 250px; border-radius: 10px; margin: 3px;
         background:blue; }
  img{ width: 350px; height: 200px; border-radius: 10px;}
  input { width: 400px; border-radius: 2px; padding: 3px; margin: 2px;}
 </style></head>
<body><div><input type="button" value="以差值模式混合图像和颜色" id=
"myBtnDifference"/></div>
<div class="box"><img src="img/B156B.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("mix-blend-mode","difference")表示以 difference 模式混合图像和颜色。在 CSS3 中 mix-blend-mode 属性主要用于源与背景颜色或背景图像的混合，目前浏览器支持 16 种混合模式，即 normal、multiply、screen、overlay、darken、lighten、color-dodge、color-burn、hard-light、soft-light、difference、exclusion、hue、saturation、color、luminosity，其中每一种混合模式都有其各自的计算模式。各混合模式的功能如下。

(1) normal：正常模式，此模式将混合色的像素通过所用的颜色显示出来。

(2) multiply：正片叠底模式，此模式将查看每个通道中的颜色信息，并将基色与混合色复合。

(3) screen：滤色模式，滤色模式与正片叠底模式正好相反，它将图像的基色与混合色结合起来产生比两种颜色都浅的第 3 种颜色。

(4) overlay：叠加模式，叠加模式把图像的基色与混合色相混合产生一种中间色。

(5) darken：变暗模式，此模式将查看每个通道的颜色信息，并选择基色或混合色中较暗的颜色作为结果色。

(6) lighten：变亮模式，此模式将查看每个通道的颜色信息，并选择基色或混合色中较亮的颜色作为结果色。

(7) color-dodge：颜色减淡模式，此模式将查看每个通道的颜色信息，并通过减小对比度使基色变亮以反映混合色，与黑色混合不会发生变化。

(8) color-burn：颜色加深模式，此模式将查看每个通道的颜色信息，并通过增加对比度使基色变暗以反映混合色，与白色混合不会产生变化。

(9) hard-light：强光模式，强光模式将产生一种强光照射的效果。如果混合色的颜色比基色的颜色的像素更亮一些，那么结果色的颜色将更亮；如果混合色的颜色比基色的颜色的像素更暗一些，那么结果色将更暗。

(10) soft-light：柔光模式，柔光模式会产生一种柔光照射的效果。如果混合色的颜色比基色的颜色的像素更亮一些，那么结果色将更亮；如果混合色的颜色比基色的颜色的像素更暗一些，那么结果色的颜色将更暗，使图像的亮度反差增大。

(11) difference：差值模式，此模式将查看每个通道的颜色信息，差值模式将从图像中基色的颜色的亮度值减去混合色的颜色的亮度值，如果结果为负，则取正值，产生反相效果。

(12) exclusion：排除模式，排除模式与差值模式相似，但是具有高对比度和低饱和度的特点，比用差值模式获得的颜色要柔和、明亮一些。

(13) hue：色相模式，色相模式只用混合色的颜色的色相值进行着色，而使饱和度和亮度值保持不变。

(14) saturation：饱和度模式，饱和度模式的作用方式与色相模式相似，它只用混合色的颜色的饱和度值进行着色，而使色相值和亮度值保持不变。

(15) color：颜色模式，颜色模式能够使用混合色的颜色的饱和度值和色相值同时进行着色，而使基色的颜色的亮度值保持不变。颜色模式可以看成是饱和度模式和色相模式的综合效果。

(16) luminosity：亮度模式，亮度模式能够使用混合色的颜色的亮度值进行着色，而保持基色的颜色的饱和度值和色相值不变，其实就是用基色中的色相和饱和度以及混合色的亮度创建结果色。

此实例的源文件名是 myHtmlB159.html。

133 综合两种滤镜实现以 X 光效果显示图像

此实例主要通过使用 grayscale()和 invert()两种滤镜方法实现以 X 光效果显示图像。当在 Google Chrome 浏览器中显示该页面时将显示未使用滤镜处理的图像；单击“灰度滤镜”按钮，图像在使用灰度滤镜处理后的效果如图 133-1 所示；单击“反色滤镜”按钮，图像在使用反色滤镜处理后的效果如图 133-2 所示；单击“X 光滤镜”按钮，图像在同时使用灰度滤镜和反色滤镜处理后的效果如图 133-3 所示。有关此实例的主要代码如下。



图 133-1



图 133-2



图 133-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnGray").click(function() {      //灰度滤镜
    $("img").css("-webkit-filter", "grayscale(100%) ");
   });
   $("#myBtnInvert").click(function() {     //反色滤镜
    $("img").css("-webkit-filter", " invert(100%)");
   });
   $("#myBtnXLight").click(function() {     //X光滤镜
    $("img").css("-webkit-filter", "grayscale(100%) invert(100%)"); });});
 </script>
 <style type="text/css">
  img { width: 405px;height: 250px; border-radius: 5px; }
  input { width: 126px;border-radius: 2px;padding: 3px; margin: 2px;}
 </style></head>
<body><div><input type="button" value="灰度滤镜" id="myBtnGray"/>
     <input type="button" value="反色滤镜" id="myBtnInvert"/>
     <input type="button" value="X光滤镜" id="myBtnXLight"/></div>
<img src="img/B148.jpg">
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("-webkit-filter","grayscale(100%) ")表示以灰度滤镜处理图像，grayscale(100%)方法的参数值在0～100%之间，参数值越大，灰度等级越高；$("img").css("-webkit-filter","invert(100%)")表示以反色滤镜处理图像，invert(100%)方法的参数值为0～100%，参数值越大，反色程度越高；$("img").css("-webkit-filter","grayscale(100%) invert(100%)")表示同时使用灰度滤镜和反色滤镜处理图像，从而使图像产生X光效果。

此实例的源文件名是myHtmlB148.html。

134　对元素下层的其他元素使用滤镜特效

此实例主要通过设置元素的backdrop-filter属性实现对元素下层的其他元素使用滤镜特效。当在Google Chrome浏览器中显示该页面时，单击“反相滤镜”按钮，在指定元素中使用反相滤镜对下层元素产生的特效如图134-1所示；单击“模糊滤镜”按钮，在指定元素中使用模糊滤镜对下层元素产生的特效如图134-2所示；单击“反色滤镜”按钮，在指定元素中使用反色滤镜对下层元素产生的特效如图134-3所示。有关此实例的主要代码如下。



图　134-1



图 134-2



图 134-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnHue").click(function() {      //反相滤镜
    $(".content").css("backdrop-filter", "hue-rotate(180deg)");
   });
   $("#myBtnBlur").click(function() {     //模糊滤镜
    $(".content").css("backdrop-filter", "blur(3px)");
   });
   $("#myBtnInvert").click(function() {   //反色滤镜
    $(".content").css("backdrop-filter", "invert(1)"); });});
 </script><style type="text/css">
  .backdrop { width:350px; height:200px;background: url('img/A106.jpg');}
  .content {width:350px;height:150px; position: relative;top: 25%;
            background: rgba(100,150,100,0.35); }
  h3,p{padding-left: 10px;padding-top: 10px;color: yellow;font-size: 16px;}
  input{width:108px;border-radius: 5px;margin: 2px;}
 </style></head>
<body><input type="button" value="反相滤镜" id="myBtnHue"/>
<input type="button" value="模糊滤镜" id="myBtnBlur"/>
<input type="button" value="反色滤镜" id="myBtnInvert"/>
<div class="backdrop"><div class="content">
  <h3>罗帅 等编著</h3><p>HTML5 炫酷实例集锦</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，backdrop-filter 属性用于对添加此滤镜的元素的下层元素实现滤镜特效，而 filter 属性只能对添加此滤镜的元素本身生效，无法应用到其下层元素。

backdrop-filter 可以使用的滤镜主要如下。

(1) blur()滤镜：实现高斯模糊。

(2) brightness()滤镜：实现亮度调整。

(3) contrast()滤镜：实现对比度调整。

(4) drop-shadow()滤镜：实现阴影特效。

(5) grayscale()滤镜：实现灰度特效。

(6) hue-rotate()滤镜：实现色相旋转。

(7) invert()滤镜：实现反色。

(8) opacity()滤镜：实现透明度调整。

(9) sepia()滤镜：实现褐色复古特效。

(10) saturate()滤镜：调整饱和度。

此实例的源文件名是 myHtmlB131. html。

135 使用 alpha 通道对元素实现半透明显示

此实例主要通过设置元素的背景颜色的 alpha 通道实现以半透明的效果显示文本。当在 Google Chrome 浏览器中显示该页面时，单击“不透明显示”按钮，文本的不透明显示效果如图 135-1 所示；单击“半透明显示”按钮，文本的半透明显示效果如图 135-2 所示。有关此实例的主要代码如下。



图 135-1



图 135-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnColor").click(function() {        //不透明显示
    $("p").css("background-color", "rgb(93,91,205)");
   });
   $("#myBtnAlpha").click(function() {        //半透明显示
    $("p").css("background-color", "rgba(93,91,205,0.5)"); });});
 </script><style>
  .box { background: #0099CC; width: 400px; height: 250px;
         background-image: url(img/B106.jpg);
         background-repeat: no-repeat;
         background-size: 100% 100%; display: flex;
         justify-content: center;align-items: center;
         border-radius: 5px;}
  p {text-indent: 35px; width: 300px; color: yellow;
     border-radius: 5px; padding: 20px; margin: 30px;}
  input {width: 193px;border-radius: 2px;margin: 2px;}
 </style></head>
<body><input type="button" value="不透明显示" id="myBtnColor"/>
<input type="button" value="半透明显示" id="myBtnAlpha"/>
```

```
<div class="box"><p>朝天门位于重庆城东北长江、嘉陵江交汇处,襟带两江; 壁垒三面,地势中高,两侧渐次向下倾斜,人行石阶沿山而上。明初戴鼎扩建重庆旧城,按九宫八卦之数造城门 17 座,其中规模最大的一座城门即朝天门。门上原书四个大字"古渝雄关"。朝天门,为历代官接皇帝圣旨的地方,因古代称皇帝为天子,故此而得名。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("p").css("background-color","rgba(93,91,205,0.5)")用于设置p元素的背景颜色的透明度为0.5,即半透明。在CSS3中,可以通过对RGB颜色设定alpha通道的方法来定义RGBA颜色。所谓RGBA颜色,是指使用红色值(R)、绿色值(G)、蓝色值(B)、alpha通道值(A)来定义的颜色。其中,alpha通道值的范围为0～1.0,0表示完全透明,1表示不透明。

此实例的源文件名是myHtmlB106.html。

136 通过设置HSLA实现元素的半透明显示

此实例主要通过对元素背景的HSLA颜色设置alpha通道实现以半透明的效果显示文本。当在Google Chrome浏览器中显示该页面时,单击"不透明显示"按钮,文本的不透明显示效果如图136-1所示;单击"半透明显示"按钮,文本的半透明显示效果如图136-2所示。有关此实例的主要代码如下。

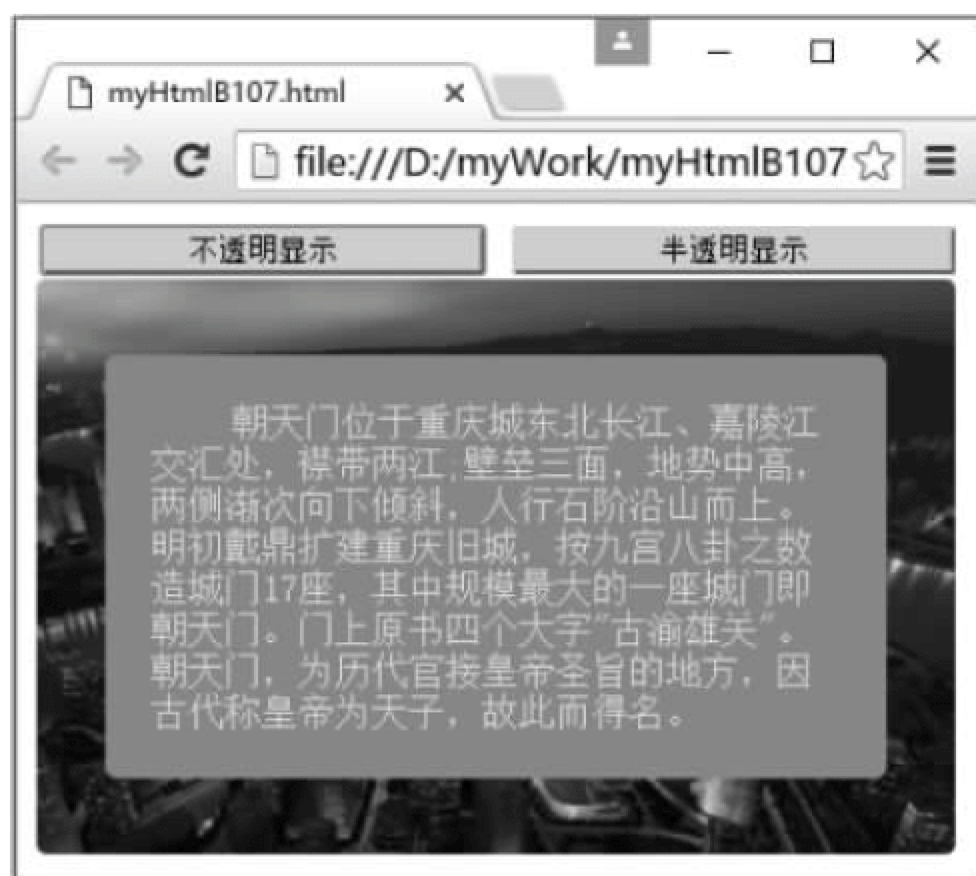


图 136-1



图 136-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnColor").click(function() {        //不透明显示
    $("p").css("background-color", "hsl(120,100%,50%)");
   //$("p").css("color", "hsl(0,100%,100%)");
  });
   $("#myBtnHsla").click(function() {        //半透明显示
    $("p").css("background-color", "hsla(120,100%,50%,0.5)"); });});
 </script>
 <style>
  .box { background: #0099CC;width: 400px; height: 250px;
         background-image: url(img/B106.jpg);
```

```
        background-repeat: no-repeat;background-size: 100% 100%;
        display: flex;justify-content: center;
        align-items: center;border-radius: 5px;}
  p {text-indent: 35px;width: 300px; color: yellow;
     border-radius: 5px;padding: 20px;margin: 30px;}
 input {width: 193px;border-radius: 2px;margin: 2px;}
</style></head>
<body><input type="button" value="不透明显示" id="myBtnColor"/>
<input type="button" value="半透明显示" id="myBtnHsla"/>
<div class="box"><p>朝天门位于重庆城东北长江、嘉陵江交汇处,襟带两江;壁垒三面,地势中高,两侧渐次
向下倾斜,人行石阶沿山而上。明初戴鼎扩建重庆旧城,按九宫八卦之数造城门 17 座,其中规模最大的一座城
门即朝天门。门上原书四个大字"古渝雄关"。朝天门,为历代官接皇帝圣旨的地方,因古代称皇帝为天子,故此
而得名。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("p").css("background-color","hsla(120,100%,50%,0.5)")用于以 HSLA 颜色的形式设置半透明的 p 元素背景颜色。在 CSS3 中除了可以使用 RGB 颜色外,还可以使用 HSL 颜色。HSL 颜色使用色调(H)、饱和度(S)、亮度(L)来定义颜色。其中,色调值用 0 或 360 表示红色、120 表示绿色、240 表示蓝色。当色调值大于 360 时(如 480),则实际值等于 480 除以 360 之后的余数 120(取模运算)。饱和度和亮度的取值范围均为 0%到 100%。用户可以通过对 HSL 颜色设定 alpha 通道的方法来定义 HSLA 颜色。HSLA 颜色是使用色调(H)、饱和度(S)、亮度(L)、alpha 通道值(A)来定义颜色,alpha 通道值的范围为 0~1.0,0 表示完全透明,1 表示不透明。

此实例的源文件名是 myHtmlB107.html。

137 以多种混合模式处理图像和文字颜色

此实例主要通过设置元素的 mix-blend-mode 属性实现以多种混合模式将容器的背景图像和文字颜色混合以生成特效。当在 Google Chrome 浏览器中显示该页面时将显示默认的背景图像和文字,单击“正片叠底模式”按钮,背景图像和文字颜色经过正片叠底模式混合后的效果如图 137-1 所示;单击“排除模式”按钮,背景图像和文字颜色经过排除模式混合后的效果如图 137-2 所示;单击“亮度模式”按钮,背景图像和文字颜色经过亮度模式混合后的效果如图 137-3 所示;单击“强光模式”按钮,背景图像和文字颜色经过强光模式混合后的效果如图 137-4 所示。有关此实例的主要代码如下。



图 137-1



图 137-2



图 137-3



图 137-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnMultiply").click(function() {       //正片叠底模式
    $("p").css("mix-blend-mode", "multiply"); });
   $("#myBtnExclusion").click(function() {      //排除模式
    $("p").css("mix-blend-mode", "exclusion"); });
   $("#myBtnLuminosity").click(function() {     //亮度模式
    $("p").css("mix-blend-mode", "luminosity"); });
   $("#myBtnHard-light").click(function() {     //强光模式
    $("p").css("mix-blend-mode", "hard-light"); });});
 </script>
 <style type="text/css">
   p { -webkit-text-stroke: 1px green;font-size: 90px; margin: 30px; color: green;
       text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.15), 2px 2px 1px rgba(0, 0, 0, 0.25), 3px 3px 1px rgba(0,
0, 0, 0.35), 4px 4px 1px rgba(0, 0, 0, 0.5);
       font-weight: bold; margin: 0;text-transform: capitalize;
       text-align: center;line-height: 250px; }
  .box { width: 400px;height: 250px;border-radius: 10px; margin: 3px;
         background: url(img/B158B.jpg) no-repeat center; }
  input { width: 92px; border-radius: 2px; padding: 3px;margin: 1px;}
 </style></head>
<body><div>
     <input type="button" value="正片叠底模式" id="myBtnMultiply"/>
     <input type="button" value="排除模式" id="myBtnExclusion"/>
     <input type="button" value="亮度模式" id="myBtnLuminosity"/>
     <input type="button" value="强光模式" id="myBtnHard-light"/></div>
<div class="box"><p>炫酷实例</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("p").css("mix-blend-mode","multiply")表示以正片叠底模式混合容器的背景图像和文字颜色；$("p").css("mix-blend-mode","exclusion")表示以排除模式混合容器的背景图像和文字颜色；$("p").css("mix-blend-mode","luminosity")表示以亮度模式混合容器的背景图像和文字颜色；$("p").css("mix-blend-mode","hard-light")表示以强光模式混合容器的背景图像和文字颜色。在 CSS3 中，mix-blend-mode 属性主要用于源与背景颜色或背景图像的混合。目前浏览器支持 16 种混合模式，即 normal、multiply、

screen、overlay、darken、lighten、color-dodge、color-burn、hard-light、soft-light、difference、exclusion、hue、saturation、color、luminosity，其中每一种混合模式都有各自的计算模式。

此实例的源文件名是 myHtmlB160.html。

138　使用对比度和模糊滤镜实现相互粘滞

此实例主要通过使用对比度滤镜和模糊滤镜实现两球相遇时的粘滞特效。当在 Google Chrome 浏览器中显示该页面时，在黑球向左、蓝球向右相背离开之前，相互之间会产生类似于拖尾的难分难舍的粘滞效果，如图 138-1 所示；在黑球向右、蓝球向左相背离开之前，相互之间也会产生类似于拖尾的难分难舍的粘滞效果，如图 138-2 所示。有关此实例的主要代码如下。

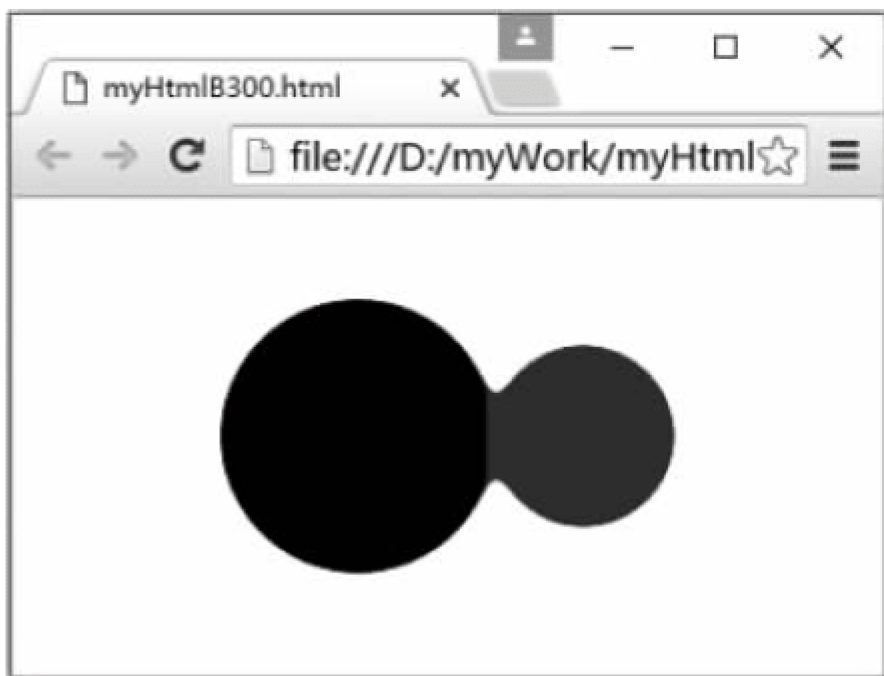


图　138-1

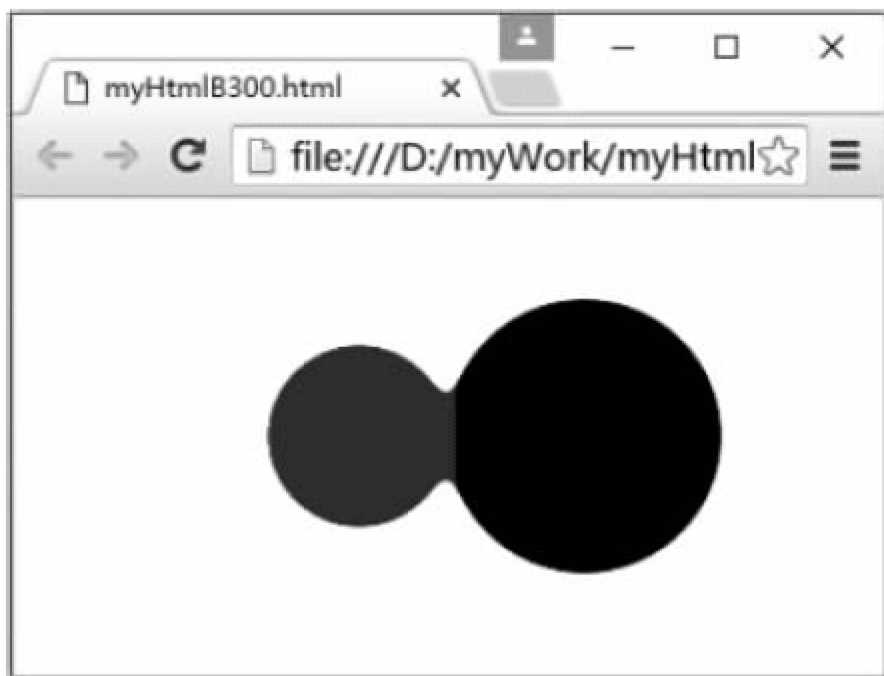


图　138-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myFilter{ position: absolute; top: 50%;left: 50%;transform: translate(-50%, -50%); width:
300px;height: 200px; -webkit-filter:contrast(20);background:white;}
  /*绘制黑球*/
  .myFilter::before{ content:""; position: absolute; border-radius: 50%; width:120px; height:120px;
background:black; top:40px;left:0px;z-index:2; -webkit-filter:blur(6px); animation:moveLeft 20s
ease-out infinite; }
  /*绘制蓝球*/
  .myFilter::after{ content:""; position: absolute;width:80px; height:80px; border-radius: 50%;
background:blue; top:60px; right:0px; z-index:2; -webkit-filter:blur(6px); animation:moveRight 20s
ease-out infinite; }
  @keyframes moveLeft{ 50%{ left:200px; } }          /*左移关键帧位置*/
  @keyframes moveRight{ 50%{ right:200px; } }        /*右移关键帧位置*/
</style></head>
<body><div class="myFilter"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，contrast()用于设置对象的对比度，如果在.myFilter{}中省略-webkit-filter：contrast(20)这行代码，则运行效果如图 138-3 所示；blur()用于设置对象的模糊程度，如果在.myFilter::before{}和.myFilter::after{}中省略-webkit-filter：blur(6px)这行代码，则运行效果如图 138-4 所示。

此实例的源文件名是 myHtmlB300.html。

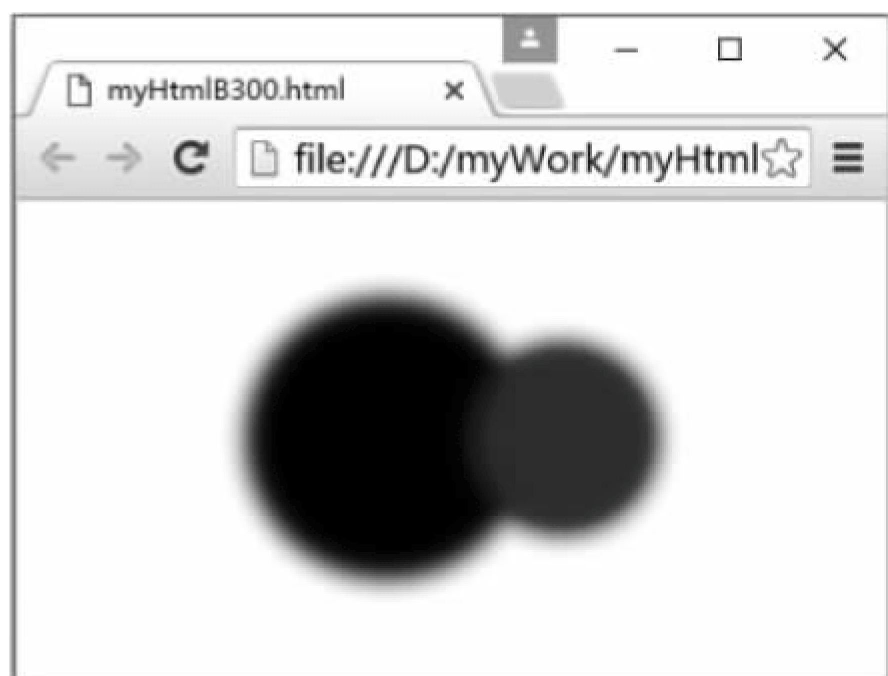


图 138-3

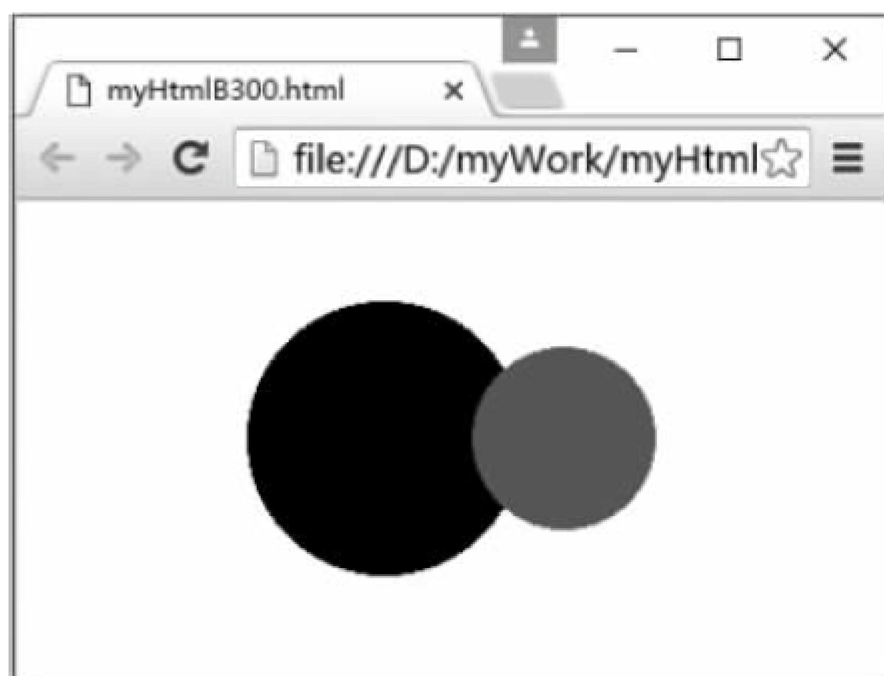


图 138-4

139 使用滤镜对不规则图像轮廓添加阴影

此实例主要通过使用 drop-shadow()实现阴影沿着不规则的图像轮廓环绕。当在 Google Chrome 浏览器中显示该页面时，单击“以正常效果显示图像”按钮将显示 png 图像在未添加阴影前的效果；单击“以阴影效果显示图像”按钮，png 图像添加阴影后的效果如图 139-1 所示。有关此实例的主要代码如下。



图 139-1



图 139-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function(){
   $("#myBtnnormal").click(function() {       //以正常效果显示图像
    $("img").css("-webkit-filter", "drop-shadow(0px 0px 0px black)"); });
   $("#myBtndrop").click(function() {         //以阴影效果显示图像
    $("img").css("-webkit-filter", "drop-shadow(5px 5px 10px black)"); });});
 </script>
 <style type="text/css">
   img { -webkit-filter: drop-shadow(5px 5px 10px black); }
   button { width: 190px; }
```

```
    div { width: 400px; margin: 20px auto; text-align: center; }
  </style></head>
<body><p><button id="myBtnnormal">以正常效果显示图像</button>
    <button id="myBtndrop">以阴影效果显示图像</button></p>
<div><img src="img/B302.png"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-filter:drop-shadow(5px 5px 10px black)用于对元素（图像）添加阴影，就好像太阳照在图像上使其产生影子一样。drop-shadow()的参数格式如下。

```
drop-shadow(x 偏移量, y 偏移量,模糊半径,颜色值)
```

在 CSS3 中，大多数时候可以使用 box-shadow 属性来实现阴影，如果对此实例中的图像使用 box-shadow：5px 5px 10px black 来实现阴影，则会产生如图 139-2 所示的阴影，这可能有违初衷。

此实例的源文件名是 myHtmlB302.html。

140　使用 skew()方法模拟 3D 风格的阴影

此实例主要在 after 选择器中使用 skew()方法倾斜阴影，从而使阴影呈现 3D 投影的特殊效果。当在 Google Chrome 浏览器中显示该页面时，在图像的背后将呈现一块倾斜的阴影，如同阳光照射在展板图像上产生的阴影一样，如图 140-1 所示。有关此实例的主要代码如下。



图　140-1

```
<!doctype html><html><head><meta charset="UTF-8">
  <style type="text/css">
   .myBox { top: 50%;left: 50%;transform: translate(-50%, -50%);
          position: absolute;padding: 2px; background: white;border-radius: 5px;
          box-shadow: 1px 2px 4px rgba(0, 0, 0, 0.5); }
   .myBox img { border-radius: 5px; }
   .myBox:after { content: ''; position: absolute; width:100px; height: 80px;
             box-shadow: 100px 0 10px rgba(0, 0, 0, 0.2);
             bottom: 0;right:65px;z-index: -1;transform: skew(-40deg); }
```

```
</style></head>
<body><div class="myBox"><img src='img/B268A.jpg'/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：100px 0 10px rgba(0，0，0，0.2)用于产生阴影，100px 表示阴影偏离 X 轴的距离，10px 表示阴影半径，0.2 表示阴影的透明度；transform：skew(－40deg)用于将阴影逆时针倾斜 40°，如果省略此行代码，则阴影将以矩形效果显示。

此实例的源文件名是 myHtmlB303.html。

141 在圆角图像下方添加渐变的倒影图像

此实例主要通过设置元素的 box-reflect 属性实现在圆角图像的下方添加一个渐变的倒影图像。当在 Google Chrome 浏览器中显示该页面时，单击“添加渐变的倒影图像”按钮，将在该图像的下方添加一个渐变的倒影图像，如图 141-1 所示。有关此实例的主要代码如下。



图 141-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnReflect").click(function() {        //添加渐变的倒影图像
    $("img").css("-webkit-box-reflect", "below 0px linear-gradient(transparent, white)"); });});
 </script>
 <style>
  img { border-top-left-radius: 5px; border-top-right-radius: 5px;}
  button{ width: 448px; margin: 1px; }
 </style></head>
<body><div><button id="myBtnReflect">添加渐变的倒影图像</button></div>
<img src="img/B190.jpg"></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("-webkit-box-reflect","below　0px linear-gradient(transparent,white)")用于在$("img")元素的下方以0px间隔距离添加渐变的倒影图像。在CSS3中，box-reflect属性用于设置或检索对象的倒影，该属性的语法格式如下。

```
box-reflect: none | <direction><offset>? <mask-box-image>?
```

其中，none是默认值，表示无倒影。<direction>有4个属性值，above指定倒影在对象的上边；below指定倒影在对象的下边；left指定倒影在对象的左边；right指定倒影在对象的右边。

<offset>表示对象与倒影之间的间隔，有两种表示方式，<length>指用长度值来定义倒影与对象之间的间隔，可以为负值；<percentage>指用百分比来定义倒影与对象之间的间隔，可以为负值。

<mask-box-image>表示倒影上的遮罩层，有6种形式，none指无遮罩图像；<url>指使用绝对或相对地址指定遮罩图像；<linear-gradient>指使用线性渐变创建遮罩图像；<radial-gradient>指使用径向(放射性)渐变创建遮罩图像；<repeating-linear-gradient>指使用重复的线性渐变创建遮罩图像；<repeating-radial-gradient>指使用重复的径向(放射性)渐变创建遮罩图像。

此实例的源文件名是myHtmlB190.html。

142　通过创建多层内置阴影高仿打靶图案

此实例主要设置元素的-webkit-box-shadow属性为内置阴影，从而创建类似于打靶的多层嵌套的圆环图案。当在Google Chrome浏览器中显示该页面时，单击"圆形模式"按钮，将在下面显示类似于打靶的多层嵌套的圆环图案，如图142-1所示；单击"矩形模式"按钮，将在下面显示多层嵌套的矩形图案，如图142-2所示。有关此实例的主要代码如下。



图　142-1



图　142-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
```

```
    $("#myBtnRect").click(function() {        //矩形模式
     $(".myNest").css("border-radius", "0"); });
    $("#myBtnCircle").click(function() {        //圆形模式
     $(".myNest").css("border-radius", "50%"); });});
  </script>
  <style>
   .myNest{ width: 350px; height: 350px; border-radius: 50%; background-color: orange;  -webkit-box
 -shadow: inset 0 0 0 30px brown, inset 0 0 0 60px yellow, inset 0 0 0 90px green, inset 0 0 0 120px blue,
 inset 0 0 0 150px red; }
   input{ width:165px; margin: 5px;}
  </style></head>
 <body><center><div><input type="button" value="矩形模式" id="myBtnRect"/>
          <input type="button" value="圆形模式" id="myBtnCircle"/></div>
  <div class="myNest"></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-box-shadow：inset 0 0 0 30px brown，inset 0 0 0 60px yellow，inset 0 0 0 90px green，inset 0 0 0 120px blue，inset 0 0 0 150px red 用于以棕、黄、绿、蓝、红创建多层嵌套的内置阴影。在默认情况下，box-shadow 创建的阴影是外置阴影，如果指定属性值 inset，则会创建内置阴影。

此实例的源文件名是 myHtmlB229.html。

143　通过旋转和平移阴影实现纸张的卷角、翘边

此实例主要通过使用旋转、平移、倾斜对阴影实施 2D 变换来实现纸张的卷角、翘边特效。当在 Google Chrome 浏览器中显示该页面时，纸张卷角、翘边（左上角和右下角）的效果如图 143-1 所示。有关此实例的主要代码如下。



图　143-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  * { margin: 0; padding: 0; }
  body { margin: 0; padding: 20px 30px; background-color: #F4F4F4;}
  .myBox { width: 250px; height: 350px; margin: 10px; display: inline-block;
          background: #FFF; border: 1px solid #CCC;position: relative;
          /* 设置纸张的内置阴影 */
          -webkit-Box-shadow: 0px 3px 90px rgba(0, 0, 0, 0.15) inset;
          /* 右下角翘(圆)角 */
          -webkit-border-radius: 0 0 6px 0/0 0 50px 0; }
  .myBox p { margin: 10px; padding: 50px; color: #AAA; }
  .myBox:before { content: ''; width: 50px; height: 100px; z-index: -1;
    position: absolute; bottom: 0; right: 0;
    /* 通过 transform 设置右下角的翘角阴影 */
    -webkit-Box-shadow: 20px 20px 10px rgba(0, 0, 0, 0.1);
    -webkit-transform: translate(-35px, -40px) skew(0deg, 30deg) rotate(-25deg); }
  .myBox:after { content: ''; width: 100px; height: 100px;
        top: 0; left: 0; position: absolute; display: inline-block; z-index: -1;
        /* 通过 transform 设置左上角的翘角阴影 */
        -webkit-Box-shadow: -10px -10px 10px rgba(0, 0, 0, 0.2);
        -webkit-transform: rotate(2deg) translate(20px, 25px) skew(20deg); }
 </style></head>
<body><div class="myBox"><p><img src="img/B306.jpg"></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：rotate(2deg) translate(20px，25px) skew(20deg)用于将产生的阴影旋转2°，平移到参数指定的位置(20px，25px)，然后倾斜20°。在CSS3中，transform属性用于对元素应用2D或3D转换，该属性允许对元素进行旋转、缩放、移动或倾斜。在一般情况下，transform属性值通常指定一种操作，例如rotate(2deg)，也可以像实例这样并列指定多种操作方法，这些方法包括matrix(n,n,n,n,n,n)、matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n, n,n)、translate(x,y)、translate3d(x,y,z)、translateX(x)、translateY(y)、scale(x,y)、translateZ(z)、scale3d(x,y,z)、scaleX(x)、scaleY(y)、scaleZ(z)、rotate(angle)、rotate3d(x,y,z,angle)、rotateZ(angle)、skew(x-angle,y-angle)、rotateX(angle)、rotateY(angle)、skewX(angle)、skewY(angle)、perspective(n)等。

此实例的源文件名是myHtmlB306.html。

144 以多重阴影创建 Firefox 浏览器的 logo

此实例主要通过在box-shadow属性值中指定多重阴影来创建Firefox浏览器的logo。当在Google Chrome浏览器中显示该页面时，创建的Firefox浏览器的logo如图144-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /* 创建黄色的圆饼 */
  .myFirefox{ position: absolute; top: 50%;left: 50%;
     width: 200px;height:200px;transform: translate(-50%, -50%);
     border:8px solid #EEE; border-radius:50%; background:#F48D00;
     overflow:hidden; box-shadow:0 0 2px 0 #999; }
  /* 通过多重阴影产生蓝色的异形 */
```

```
  .myFirefox::before{ content: ""; width: 46px; height: 46px;
    background: lightblue; border-radius: 100%; margin: auto;
    position: absolute;top: 90px;left: 50px; border: 1px solid lightblue;
    box-shadow: -10px 0 0 5px rgba(164, 223, 239, 0.2),
      16px -60px 0 8px rgba(164, 223, 239, 0.2), -30px -50px 0 8px #F48D00,
      25px -38px 0 0px lightblue, -30px -30px 0 12px #F48D00,
      15px -60px 0 8px lightblue, 33px 31px 0 -15px lightblue,
      30px 17px 0 -13px #F48D00,20px -10px 0 0px lightblue,
      -8px -0px 0 19px #F48D00,30px 17px 0 0px lightblue,
      24px 30px 0 15px #F48D00, 20px -6px 0 28px lightblue,
      45px 10px 0 35px #F48D00, -5px -57px 0 8px #F48D00,
      20px -23px 0 45px lightblue, -5px -80px 0 8px lightblue,
      -30px -70px 0 8px #F48D00, -5px -57px 0 8px #F48D00,
      43px -11px 0 55px #F48D00, 22px -39px 0 55px lightblue; }
 </style></head>
<body><div class="myFirefox"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，.myFirefox{width: 200px;height: 200px;border:8px solid #EEE;border-radius:50%;background: #F48D00;box-shadow:0 0 2px 0 #999;}用于创建黄色背景的圆饼，如图 144-2 所示；.myFirefox::before{}中的 box-shadow 用于创建多个异形的图案，如图 144-1 所示。在 CSS3 中，box-shadow 属性用于向指定的元素添加一个或多个阴影，该属性由逗号分隔的阴影列表，每个阴影由 2～4 个长度值、可选的颜色值以及可选的 inset 关键字来规定，省略长度的值是 0。

此实例的源文件名是 myHtmlB297.html。

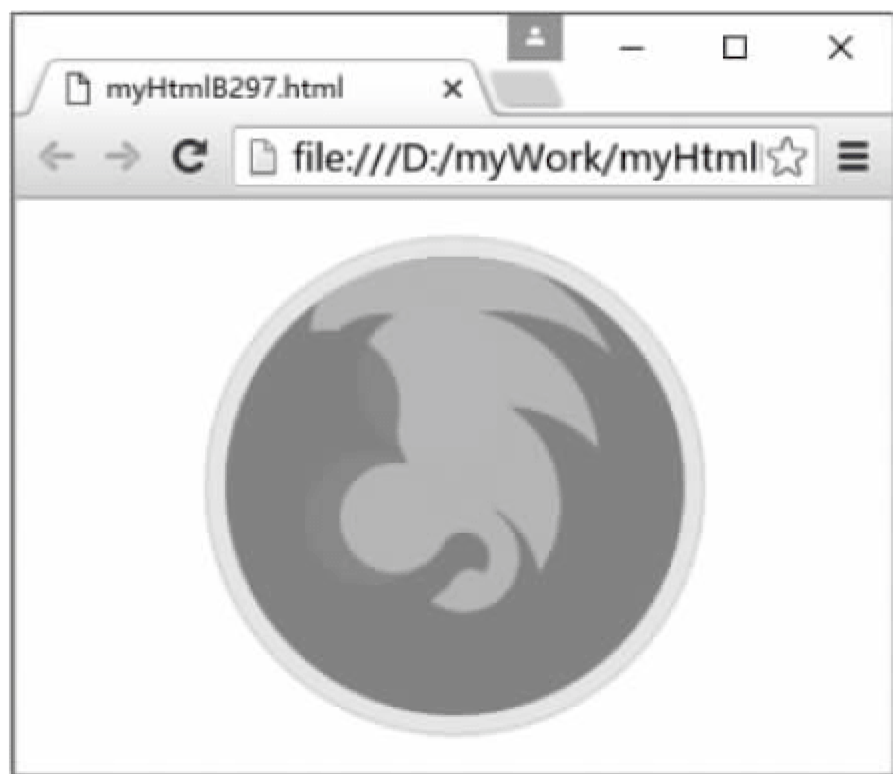


图 144-1

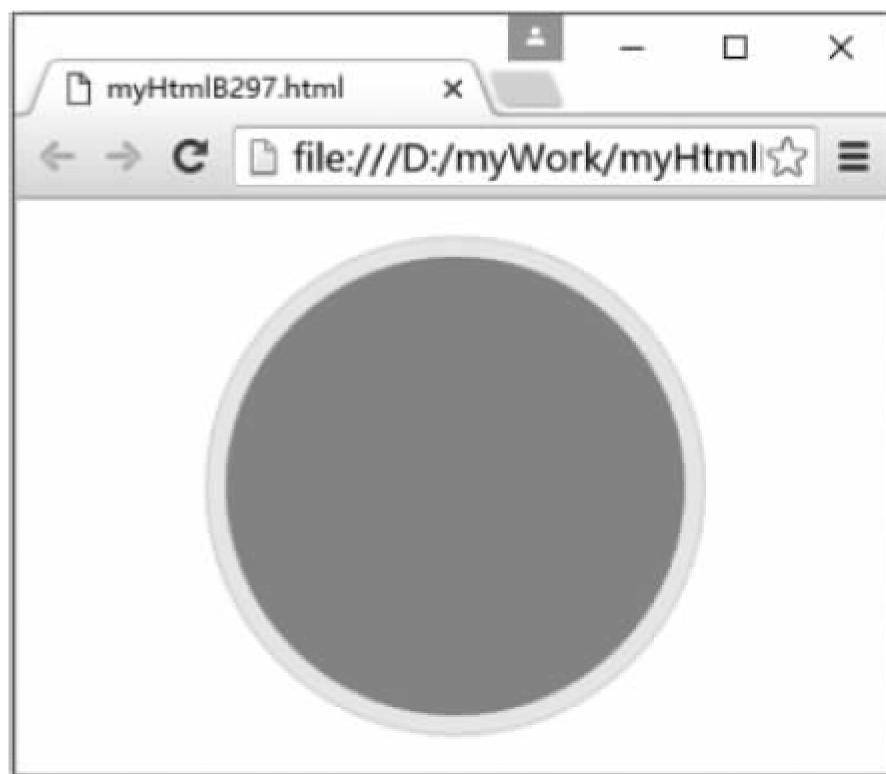


图 144-2

145 通过选择器创建 Sogou 浏览器的 logo

此实例主要使用 before 选择器向指定元素(圆饼)中插入文字“S”，从而创建 Sogou 浏览器的 logo。当在 Google Chrome 浏览器中显示该页面时，创建的 Sogou 浏览器的 logo 如图 145-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
/ * 创建蓝色的圆饼 * /
.mySogou{ position: absolute;top: 50 % ;left: 50 % ;width: 200px;height: 200px;
         transform: translate( - 50 % , - 50 % ); border: 8px solid white;
         box - shadow: 2px 1px 2px 2px gray;border - radius: 50 % ;
         overflow: hidden;background: deepskyblue;}
/ * 创建 S * /
.mySogou:before { content: "S"; position: absolute; line - height: 185px;
     top: 0;left: 0;right: 0;bottom: 0; font - weight: 600;font - size: 320px;
     color: white; font - family: 'Microsoft Yahei'; text - align: center;
    text - shadow:/ * S 左下角的浅灰色阴影 * /
                 - 5px 0px 0px #CDE4F0,
                / * S 的深蓝色阴影 * /
                0px 8px 0px #387FBF; overflow: hidden;}
</style></head>
<body><div class = "mySogou"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.mySogou{width: 200px;height: 200px;border: 8px solid white; box-shadow: 2px 1px 2px 2px gray; border-radius: 50%;background: deepskyblue;}用于创建蓝色背景的圆饼,如图 145-2 所示;.mySogou:before{content: "S"; position: absolute; line-height: 185px; top: 0;left: 0;right: 0;bottom: 0; color: white; font-family: 'Microsoft Yahei';font-weight: 600;font-size: 320px;text-align: center; text-shadow: 0px 8px 0px #387FBF; overflow: hidden;}用于在蓝色背景的圆饼上添加文字“S”,如图 145-3 所示,注意 S 的左下角和右上角的阴影都是深蓝色的,如果加上了 -5px 0px 0px #CDE4F0,则 S 的左下角的部分深蓝色阴影将变成浅灰色,如图 145-1 所示。在 CSS3 中,before 选择器用于在被选元素的内容前面插入新内容。



图 145-1

此实例的源文件名是 myHtmlB296.html。

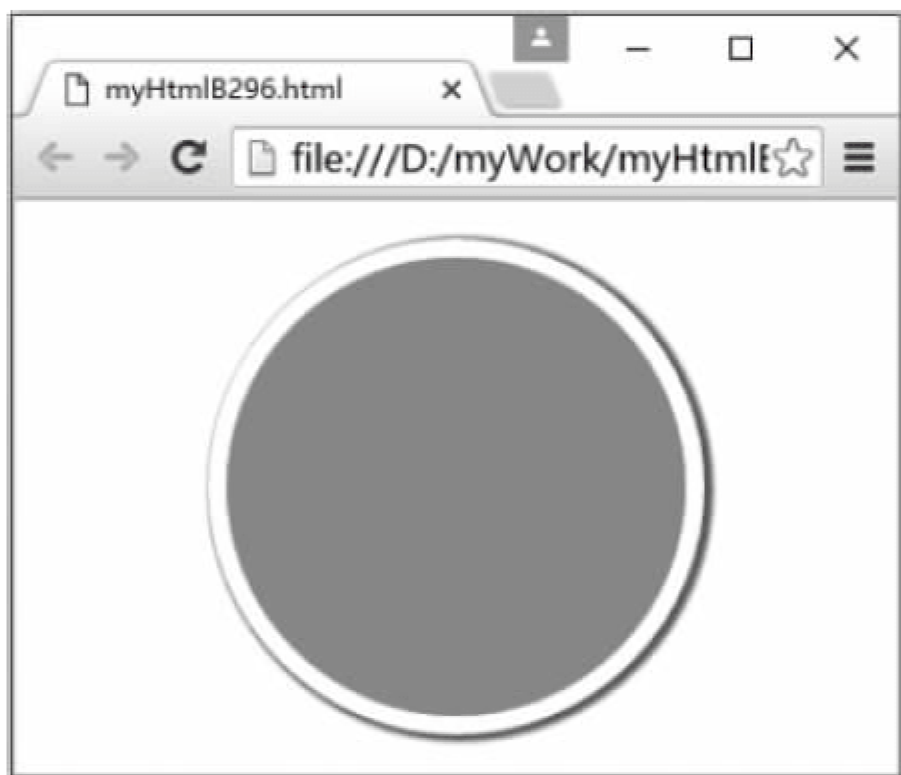


图 145-2



图 145-3

146 通过渐变创建 Safari 浏览器的 logo

此实例主要通过使用径向渐变、线性渐变以及旋转和平移等多种 CSS 方法创建 Safari 浏览器的 logo。当在 Google Chrome 浏览器中显示该页面时，创建的 Safari 浏览器的 logo 如图 146-1 所示。有关此实例的主要代码如下。

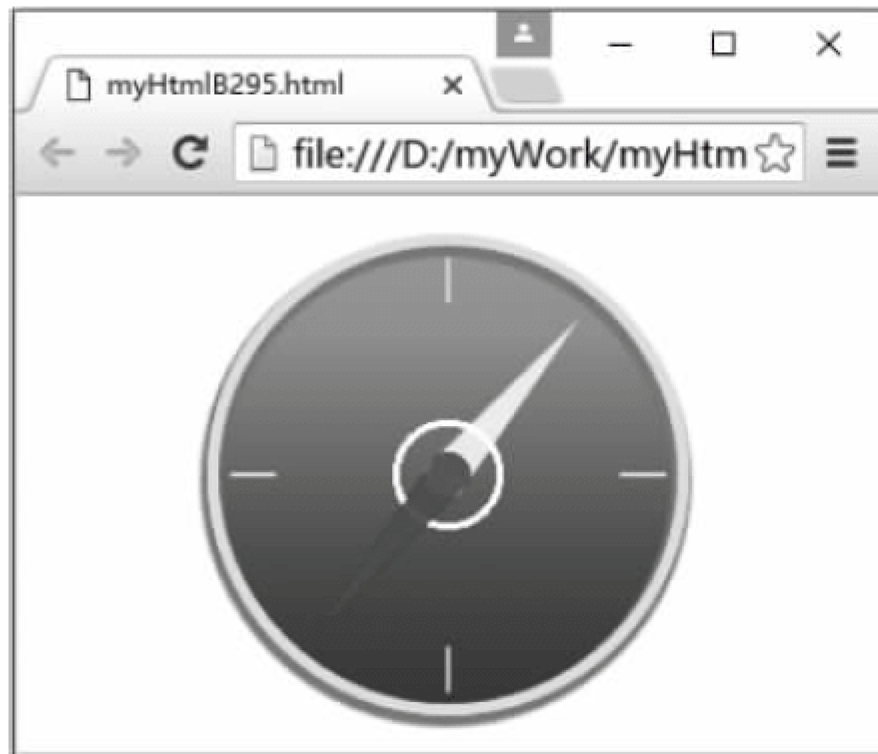


图 146-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .mySafari{ position: absolute; top: 50%;left: 50%;border-radius:50%;
     width:200px; height:200px;transform: translate(-50%, -50%);
     /*最外层的灰色圆环*/
     border:5px solid #E8E8E8;
     box-shadow: -1px 3px 1px 2px #999,        /*最外层的灰色圆环下面的外阴影*/
                 -1px 3px 1px 2px #999 inset;  /*最外层的灰色圆环上面的内阴影*/
    background-image:
       /*中心的白色圆环*/
       radial-gradient(transparent 30%,white 30%,white 34%, transparent 34%),
       /*3 点钟指示线*/
       linear-gradient(rgba(255,255,255,0.8) 100%, transparent 100%),
       /*9 点钟指示线*/
       linear-gradient(rgba(255,255,255,0.8) 100%, transparent 100%),
       /*6 点钟指示线*/
       linear-gradient(rgba(255,255,255,0.8) 100%, transparent 100%),
       /*12 点钟指示线*/
       linear-gradient(rgba(255,255,255,0.8) 100%, transparent 100%),
       /*产生圆环内部的蓝色线性渐变背景 */
       linear-gradient(#1DE3FF 0%, #1F52EF 100%);
       /*6 组尺寸必须与 background-image 中的 6 个背景图像完全对应*/
       background-size: 50% 50%,20px 2px,20px 2px,2px 20px,2px 20px,100%,100%;
       background-repeat:no-repeat;
       /*6 组位置坐标必须与 background-image 中的 6 个背景图像完全对应*/
       background-position:center center, 175px center,5px center,center 175px,center 5px,0 0;}
  .mySafari::before{ content:""; position: absolute; z-index:-1;
     top: 10px;left: 50%; border-radius:10px;
     /*通过切分边框线的策略创建白色的三角形指针*/
     border-bottom:100px solid rgba(255,255,255,0.9);
     border-left:10px solid transparent; border-right:10px solid transparent;
```

```
        transform - origin:center 90px;
        /* 旋转和平移白色的三角形指针指向右上角 */
        transform:translate( - 50 % , 0 % ) rotate(40deg); }
    .mySafari::after{ content:""; position: absolute;
                       top: 10px;left: 50 % ; border - radius:10px;
                       /* 通过切分边框线的策略创建红色的三角形指针 */
                       border - bottom:100px solid rgba(255,0,0,0.9);
                       border - left:10px solid transparent;
                       border - right:10px solid transparent;
                       transform - origin:center 90px;
                       /* 旋转和平移红色的三角形指针指向左下角 */
                       transform:translate( - 50 % , 0 % ) rotate(220deg);}
    /* 当鼠标指针悬浮时旋转白色的三角形指针 */
    .mySafari:hover::before{transition:transform 1s;
                            transform:translate( - 50 % , 0 % ) rotate(70deg);}
    /* 当鼠标指针悬浮时旋转红色的三角形指针 */
    .mySafari:hover::after{transition:transform 1s;
                           transform:translate( - 50 % , 0 % ) rotate(250deg);}
  </style></head>
<body><div class = "mySafari"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，.mySafari{width:200px; height:200px; border-radius:50%; border:5px solid #E8E8E8; box-shadow: -1px 3px 1px 2px #999, -1px 3px 1px 2px #999 inset; background-image:radial-gradient(transparent 30%,white 30%, white 34%, transparent 34%),linear-gradient(#1DE3FF 0%, #1F52EF 100%);background-size:50% 50%,100%,100%;background-repeat:no-repeat;background-position:center center,0 0;}用于创建线性渐变的蓝色背景的圆盘，如图 146-2 所示；.mySafari::before{border-radius:10px;border-bottom:100px solid rgba(255,255,255,0.9); border-left:10px solid transparent;border-right:10px solid transparent;transform-origin:center 90px;transform:translate(-50%,0%) rotate(40deg);}用于创建右上角的白色指针，如图 146-3 所示；.mySafari::after{border-radius:10px; border-bottom:100px solid rgba(255,0,0,0.9);border-left:10px solid transparent;border-right:10px solid transparent; transform-origin:center 90px;transform: translate(-50%,0%) rotate(220deg);}用于创建左下角的红色指针，如图 146-1 所示。

此实例的源文件名是 myHtmlB295.html。

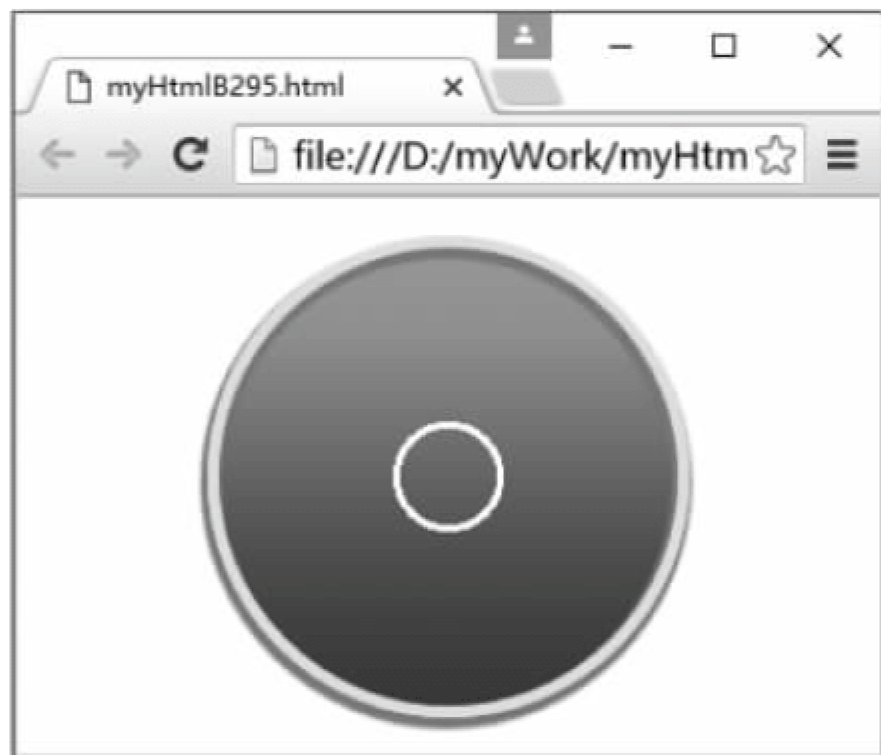


图 146-2

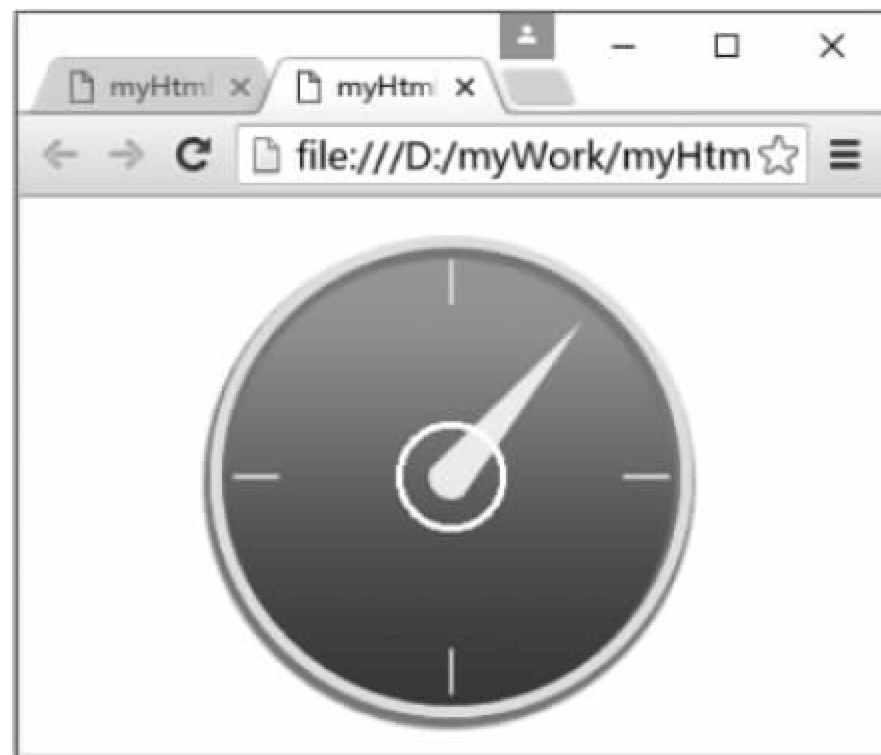


图 146-3

147　通过渐变和阴影创建 IE 浏览器的 logo

此实例主要通过径向渐变和盒子阴影创建 IE 浏览器的 logo。当在 Google Chrome 浏览器中显示该页面时，创建的 IE 浏览器的 logo 如图 147-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .myIE { margin: 10px auto; position: absolute;
          top: 50 % ;left: 50 % ;width: 200px;height: 200px;
          transform: translate( - 50 % , - 50 % );border - radius: 50 % ;
          background - image: radial - gradient(white 38 % , #09C 38 % ); }
  .myIE:before { content: ""; width: 285px;
                 height: 122px; position: absolute;
                 border - radius: 100 % ; top: 33px; left: - 45px; margin: auto;
                 box - shadow: inset 0 12px 0 13px #09C, - 35px - 8px 0 - 5px white;
                 transform: rotate( - 35deg); }
  .myIE:after { content: ""; width: 120px; height: 25px;background: #09C;
                position: absolute;top: 80px; left: 0; right: 0; margin: auto;
                box - shadow: 42px 23px 0 - 2px white; }
 </style></head>
<body><div class = "myIE"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，. myIE {width：200px；height：200px；border-radius：50%；background-image：radial-gradient(white 38%，#09C 38%)；}用于创建蓝色的圆环，如图 147-2 所示；. myIE：before {width：285px；height：122px；border-radius：100%；box-shadow：inset 0 12px 0 13px #09C，—35px —8px 0 —5px white；}用于创建水平的卫星轨迹和彗星轨迹，如图 147-3 所示，然后使用 transform：rotate(—35deg)将其逆时针旋转 35°，使其与 IE 的角度一致，如图 147-4 所示；. myIE：after { width：120px；height：25px；background：#09C；}用于实现字母 e 中间那条蓝色的水平线，box-shadow：42px 23px 0 —2px white 则以白色阴影的形式实现 e 右侧的缺口，如图 147-1 所示。

此实例的源文件名是 myHtmlB294. html。

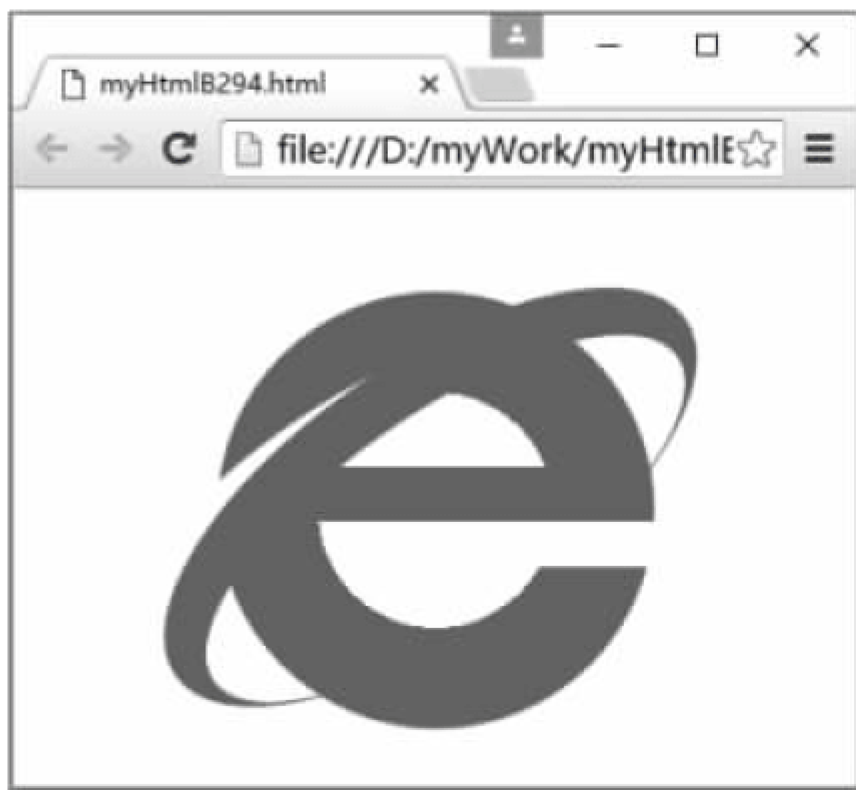


图　147-1



图　147-2

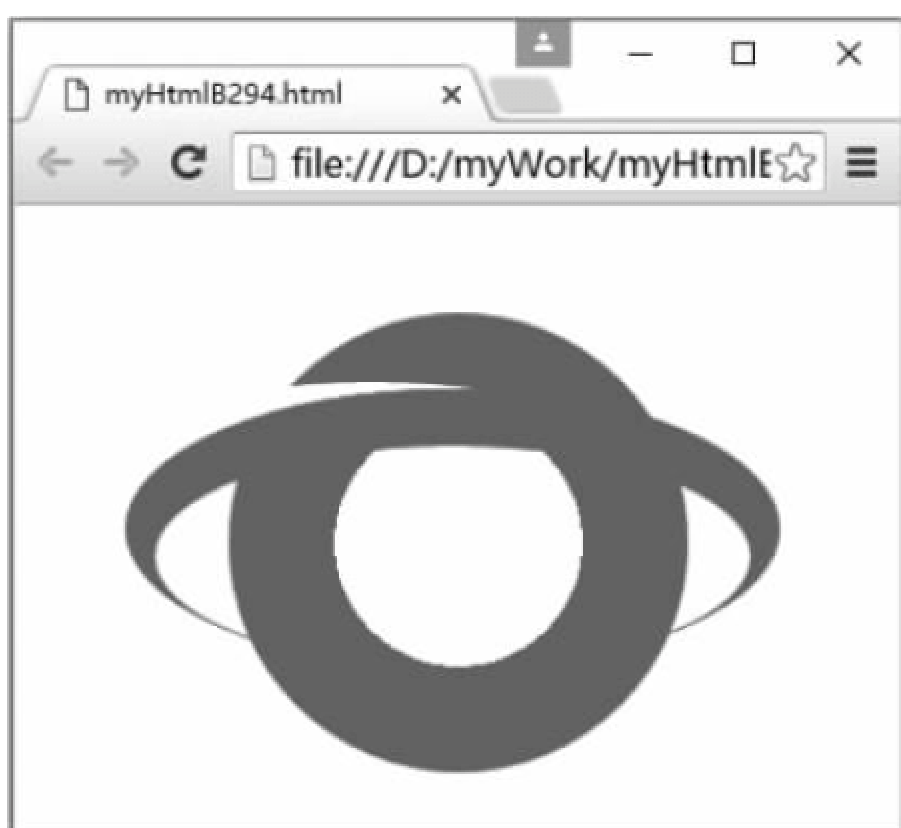


图 147-3



图 147-4

148 以椭圆叠加方式创建 Opera 浏览器的 logo

此实例主要以椭圆叠加方式绘制 Opera 浏览器的 logo。当在 Google Chrome 浏览器中显示该页面时,创建的 Opera 浏览器的 logo 如图 148-1 所示。有关此实例的主要代码如下。



图 148-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*渐变的红色椭圆*/
  .myOpera{ position: absolute;top: 50%;left: 50%;
             width: 170px;height: 200px; transform: translate(-50%, -50%);
             border-radius:50%; box-shadow:0 2px 4px black;
             background-image: linear-gradient(rgb(254, 135, 138) 0%, rgb(231, 22, 22) 50%, rgb
(128, 0, 0) 85%, rgb(184, 3, 4) 100%); }
  /*在红色椭圆里面创建白色小椭圆*/
  .myOpera:after{ content:""; position: absolute; top:50%;left:50%;
           width: 60px;height: 180px;transform: translate(-50%, -50%);
           border-radius:50%; background:white; }
  /*在红色椭圆与白色小椭圆之间嵌套半透明椭圆,突出显示叠加的椭圆边框*/
  .myOpera:before{ content:""; position: absolute; top:50%;left:50%;
```

```
            width: 72px;height: 185px; transform: translate( - 50 % , - 50 % );
            border - radius:50 % ; background:rgba(0,0,0,0.4); }
    </style></head>
    <body><div class = "myOpera"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.myOpera{width: 170px; height: 200px;border-radius:50%;background-image:linear-gradient(rgb(254,135,138) 0%, rgb(231, 22, 22) 50%, rgb(128, 0, 0) 85%, rgb(184, 3, 4) 100%);}用于创建渐变红色椭圆,如图 148-2 所示;.myOpera:after{ width: 60px;height: 180px; border-radius: 50%;background:white;}用于在红色椭圆里面创建白色小椭圆,如图 148-3 所示;.myOpera:before{width: 72px;height:185px;border-radius:50%;background: rgba(0, 0,0,0.4);}用于在红色椭圆与白色小椭圆之间嵌套半透明椭圆,突出显示叠加的椭圆边框,这主要是因为 0.4 不透明度产生的边框线效果,如图 148-1 所示。

此实例的源文件名是 myHtmlB293.html。



图 148-2



图 148-3

149 通过渐变创建 Chrome 浏览器的 logo

此实例主要通过使用 radial-gradient()方法和 linear-gradient()方法创建 Chrome 浏览器的 logo。当在 Google Chrome 浏览器中显示该页面时,创建的 Chrome 浏览器的 logo 如图 149-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .myChrome{position: absolute;top: 50 % ;left: 50 % ;width: 180px;height: 180px;
           transform: translate( - 50 % , - 50 % ); border - radius:50 % ;
           box - shadow:0 0px 4px #999,0 0 2px #DDD inset;
           background - image:
           /* 创建中心的蓝圆 */
           radial - gradient(blue 0 % ,blue 28 % , transparent 28 % ),
           /* 在蓝圆外面创建 5px 宽的白环 */
           radial - gradient(white 33 % , transparent 33 % ),
           /* 创建右下角的黄色部分 */
           linear - gradient( - 50deg,yellow 34 % , transparent 34 % ),
           /* 创建左下角的绿色部分 */
           linear - gradient(60deg,green 33 % , transparent 33 % ),
```

```
            /* 创建上端的红色部分 */
            linear-gradient(180deg,red 0%, red 30%, transparent 30%),
            /* 使用黄色补填右下角与红白相接的空白部分 */
            linear-gradient(-120deg,yellow 40%, transparent 40%),
            /* 使用绿色补填左下角与黄白相接的空白部分 */
            linear-gradient(0deg,green 45%, transparent 45%),
            /* 使用红色补填右上角与绿白相接的空白部分 */
            linear-gradient(120deg,red 50%, transparent 50%); }
</style></head>
<body><div class="myChrome"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-image：radial-gradient(blue 0%，blue 28%，transparent 28%)，radial-gradient(white 33%，transparent 33%)，linear-gradient(−50deg，yellow 34%，transparent 34%)创建的图形如图149-2所示。绿色部分和红色部分也是按照同样的思路来实现的，仅有角度和颜色值的差异。在CSS3中，radial-gradient()方法用于以径向渐变的方式创建图像，linear-gradient()方法则以线性渐变的方式创建图像。

此实例的源文件名是myHtmlB292.html。

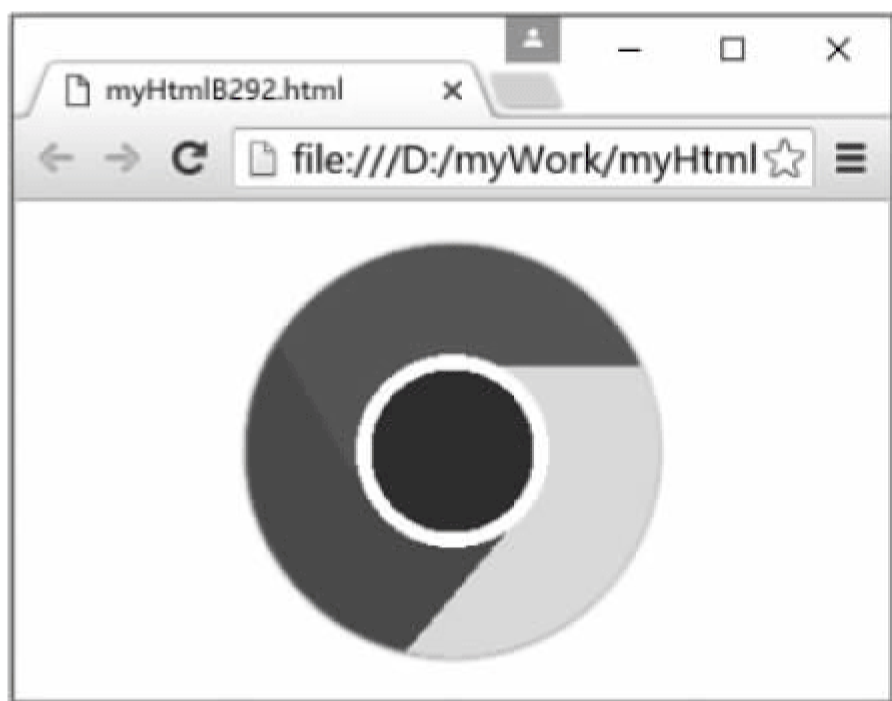


图　149-1

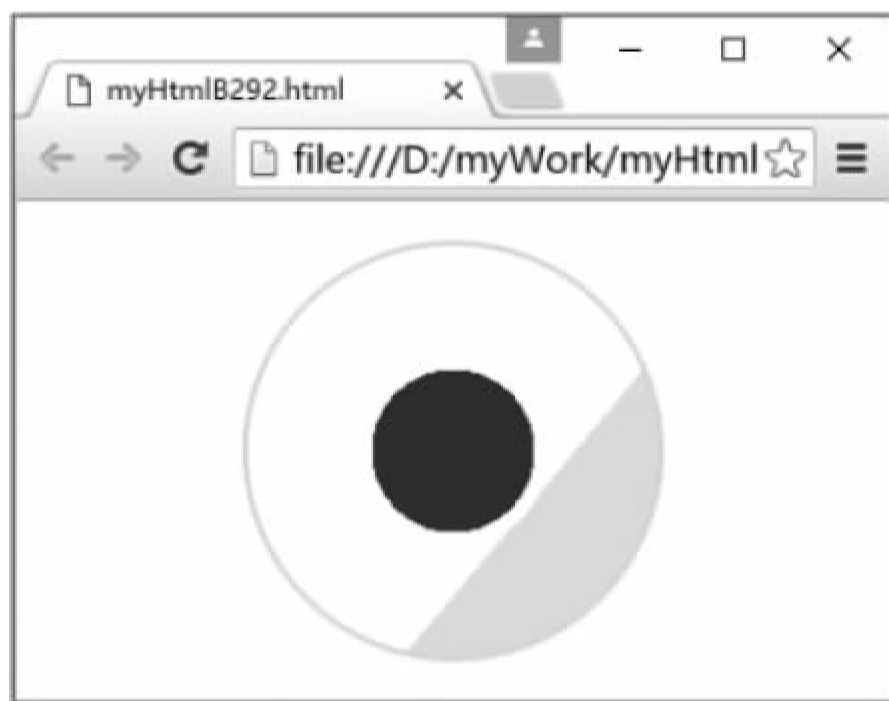


图　149-2

150　通过径向渐变和线性渐变创建穿线纽扣

此实例主要使用radial-gradient()方法创建针眼、使用linear-gradient()方法创建穿线，从而创建穿线纽扣的图案。当在Google Chrome浏览器中显示该页面时，穿线纽扣的图案如图150-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myButton{position: absolute;top: 50%;left: 50%; background: #87CEEB;
  transform: translate(-50%, -50%); border-radius: 50%; width: 180px;height:180px;
  box-shadow: inset 0 5px 5px #87CEEB, inset 0 17px 5px rgba(255,255,255,0.7),
      inset 0 -3px 3px rgba(255,255,255,0.3), inset 0 -10px 10px rgba(0,0,0,0.3),
      inset 0 -15px 10px #45B3E0, 0 7px 10px rgba(0,0,0,0.3); }
  .myButton::before{ content: ''; position:absolute;
         top: 50%;left: 50%; width: 110px;height: 106px;
         margin-left: -55px;margin-top: -55px;
         /* 绘制纽扣上的 4 个针眼 */
```

```
            background-image:
             radial-gradient(circle at 38px 38px, black 10px, transparent 10px),
             radial-gradient(circle at 73px 73px, black 10px, transparent 10px),
             radial-gradient(circle at 38px 73px, black 10px, transparent 10px),
             radial-gradient(circle at 73px 38px, black 10px, transparent 10px);
            border-radius: 50%;border-top: 1px solid rgba(0,0,0,0.6);
            border-bottom: 1px solid rgba(255,255,255,0.6);
            box-shadow:inset 0 20px 2px rgba(255,255,255,0.3),
              3px -15px 7px -4px rgba(0,0,0,0.3), 0 -14px 10px 5px #45B3E0,
               0 2px 5px 5px #87CEEB,0 10px 5px 5px rgba(255,255,255,0.6); }
   .myButton::after{ content: ''; position:absolute; top: 50%;left: 50%;
        width: 50px;height: 50px; margin-left: -25px; margin-top: -23px;
        /*绘制纽扣上的十字交叉穿线*/
        background-image:
            linear-gradient(to right, transparent 35%, #000080 35%, #4682B4 40%, #000080 45%,
  #4682B4 50%, #000080 55%, #4682B4 60%, #000080 65%, transparent 65%),
            linear-gradient(to bottom, transparent 35%, #000080 35%, #4682B4 40%, #000080 45%,
  #4682B4 50%, #000080 55%, #4682B4 60%, #000080 65%, transparent 65%);
          border-radius: 50%;
          /*将穿线旋转45°对准针眼*/
          transform: rotate(45deg);}
  </style></head>
  <body><div class="myButton"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，radial-gradient(circle at 38px 38px, black 10px, transparent 10px)表示在(38px 38px)位置创建一个半径为10px的圆，即左上角针眼，其他3个针眼的创建原理类似，只是位置不同；linear-gradient(to right, transparent 35%, #000080 35%, #4682B4 40%, #000080 45%, #4682B4 50%, #000080 55%, #4682B4 60%, #000080 65%, transparent 65%)表示按照从左到右的顺序通过线性渐变创建垂直的穿线，linear-gradient(to bottom, transparent 35%, #000080 35%, #4682B4 40%, #000080 45%, #4682B4 50%, #000080 55%, #4682B4 60%, #000080 65%, transparent 65%)表示按照从上到下的顺序通过线性渐变创建水平的穿线，如图150-2所示。最后通过transform: rotate(45deg)将刚才创建的十字交叉穿线旋转45°，从而正好落在4个针眼。在CSS3中，radial-gradient()方法用于以径向渐变的方式创建图像，linear-gradient()方法则以线性渐变的方式创建图像。

此实例的源文件名是myHtmlB291.html。



图　150-1



图　150-2

151 通过线性渐变创建充电状态的电池图案

此实例主要使用 linear-gradient()方法创建多个渐变图形,从而创建充电状态的电池图案。当在 Google Chrome 浏览器中显示该页面时,创建的充电状态的电池图案如图 151-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body{ background-color: black;}
  /*创建电池外形轮廓和充电部分的渐变图案*/
  .myBattery{ position: absolute; top: 50%;left: 50%; width: 250px;height:120px;
             transform: translate(-50%, -50%);border-radius: 10px/30px;
             /*创建电池上、下两端的边框*/
             border-left: 2px solid rgba(255,255,255,0.2);
             border-right: 2px solid rgba(255,255,255,0.2);
             background-image:
             /*创建充电及其他垂直部分的轮廓*/
             linear-gradient(to right, transparent 5%, #316D08 5%, #316D08 7%,
#60B939 8%, #60B939 10%, #51AA31 11%, #51AA31 60%, #64CE11 61%, #64CE11 63%,
#255405 63%, black 68%, black 95%, transparent 95%),
             /*创建电池水平部分的外形轮廓*/
             linear-gradient(to bottom, rgba(255,255,255,0.5) 0%, rgba(255,255, 255,0.4) 4%,
rgba(255,255,255,0.2) 7%, rgba(255,255,255,0.2) 14%, rgba(255,255, 255,0.8) 14%, rgba(255,255,
255,0.2) 40%, rgba(255,255,255,0) 41%, rgba(255,255, 255,0) 80%, rgba(255,255,255,0.2) 80%, rgba
(255,255,255,0.4) 86%, rgba(255,255, 255,0.6) 90%, rgba(255,255,255,0.1) 92%, rgba(255,255,255,
0.1) 95%, rgba(255,255, 255,0.5) 98%);}
  /*创建电池右端的电芯图案*/
  .myBattery::before{ content: ''; position:absolute; top: 32px;right: -14px;
        width: 12px;height: 55px;border-top-right-radius: 6px 10px;
        border-bottom-right-radius: 6px 10px;
        background-image: linear-gradient(to bottom, rgba(255,255, 255,0.5) 0%, rgba(255,255,255,0)
14%, rgba(255,255,255,0.8) 14%, rgba(255,255, 255,0.3) 40%, rgba(255,255,255,0) 41%, rgba(255,
255,255,0) 80%, rgba(255,255, 255,0.2) 80%, rgba(255,255,255,0.4) 86%, rgba(255,255,255,0.6)
90%, rgba(255,255, 255,0.1) 92%, rgba(255,255,255,0.1) 95%, rgba(255,255,255,0.5) 98%);}
  /*创建电池外部的透明图案*/
  .myBattery::after{ content: ''; position:absolute;width: 220px;height:120px;
         left: 10px;border-radius: 5px/30px;
         border-left: 4px solid black;border-right: 4px solid black;
         background-image: linear-gradient(to bottom, rgba(255,255,255,0.3) 4%, rgba(255,255,255,
0.2) 5%, transparent 5%, transparent 14%, rgba(255,255,255,0.3) 14%, rgba(255,255,255,0.12) 40%,
rgba(0,0,0,0.05) 42%, rgba(0,0,0,0.05) 48%, transparent 60%, transparent 80%, rgba(255,255,255,
0.3) 87%, rgba(255,255,255,0.3) 92%, transparent 92%, transparent 97%, rgba(255,255,255,0.4) 97%),
                          linear-gradient(to left, rgba(255,255,255,0.2) 0%, rgba(255,255,255,
0.2) 2%, black 2%, black 6%, transparent 6%),
                         linear-gradient(to bottom, rgba(255,255,255,0) 0%, rgba(255,255,255,0)
35%, rgba(255,255,255,0.3) 90%, rgba(255,255,255,0) 90%);}
</style></head>
<body><div class="myBattery"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.myBattery{background- image: linear-gradient(to bottom, rgba(255,255,255,0.5) 0%, rgba(255,255,255,0.4) 4%, rgba(255,

255,255,0.2) 7%, rgba(255,255,255,0.2) 14%, rgba(255,255,255,0.8) 14%, rgba(255,255,255,0.2) 40%, rgba(255,255,255,0) 41%, rgba(255,255,255,0) 80%, rgba(255,255,255,0.2) 80%, rgba(255,255,255,0.4) 86%, rgba(255,255,255,0.6) 90%, rgba(255,255,255,0.1) 92%, rgba(255,255,255,0.1) 95%, rgba(255,255,255,0.5) 98%);}中的线性渐变方法 linear-gradient()用于创建电池的水平外形轮廓,如图 151-2 所示;.myBattery{background-image:linear-gradient(to right, transparent 5%, #316D08 5%, #316D08 7%, #60B939 8%, #60B939 10%, #51AA31 11%, #51AA31 60%, #64CE11 61%, #64CE11 63%, #255405 63%, black 68%, black 95%, transparent 95%)}中的线性渐变方法 linear-gradient()用于创建电池的充电及其他垂直部分的外形轮廓,如图 151-3 所示;.myBattery::before{background-image:linear-gradient(to bottom, rgba(255,255, 255,0.5) 0%, rgba(255,255,255,0) 14%, rgba(255,255,255,0.8) 14%, rgba(255,255, 255,0.3) 40%, rgba(255,255,255,0) 41%, rgba(255,255,255,0) 80%, rgba(255,255, 255,0.2) 80%, rgba(255,255,255,0.4) 86%, rgba(255,255,255,0.6) 90%, rgba(255,255, 255,0.1) 92%, rgba(255,255,255,0.1) 95%, rgba(255,255,255,0.5) 98%);}中的线性渐变方法 linear-gradient()用于创建电池右端的电芯的外形轮廓,如图 151-4 所示;.myBattery::after{background-image:linear-gradient(to bottom, rgba(255,255, 255,0.3) 4%, rgba(255,255,255,0.2) 5%, transparent 5%, transparent 14%, rgba(255, 255,255,0.3) 14%, rgba(255,255,255,0.12) 40%, rgba(0,0,0,0.05) 42%, rgba(0,0,0, 0.05) 48%, transparent 60%, transparent 80%, rgba(255,255,255,0.3) 87%, rgba(255, 255, 255,0.3) 92%, transparent 92%, transparent 97%, rgba(255,255,255,0.4) 97%), linear-gradient(to left, rgba(255,255,255,0.2) 0%, rgba(255,255,255,0.2) 2%, black 2%, black 6%, transparent 6%),linear-gradient(to bottom, rgba(255,255,255,0) 0%, rgba(255,255,255,0) 35%, rgba(255,255,255,0.3) 90%, rgba(255,255,255,0) 90%);}中的线性渐变方法 linear-gradient()用于创建电池的透明图案,如图 151-1 所示。



图 151-1



图 151-2



图 151-3

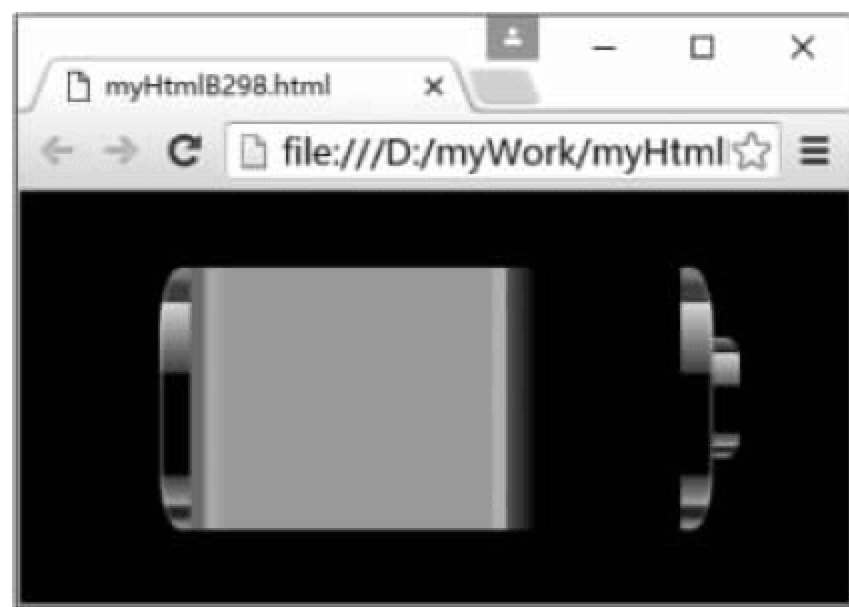


图 151-4

在 CSS3 中，linear-gradient()方法用于以线性渐变方式创建图像，在该方法中通常使用 to、top、left 这样的关键字来指示渐变方向。

此实例的源文件名是 myHtmlB298. html。

152 通过径向渐变实现邮票边缘的锯齿风格

此实例主要使用 radial-gradient()方法创建渐变背景，从而实现在一幅普通图像的边缘显示邮票边缘的锯齿风格。当在 Google Chrome 浏览器中显示该页面时，图像的边缘将呈现类似于邮票边缘的锯齿风格，如图 152-1 所示。有关此实例的主要代码如下。



图 152-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  body { background: darkgreen; text - align: center; }
  /* 设置盒子的基本样式 */
  .myStamp{ margin - top:25px; width: 280px; height: 180px;
  display: inline - block; padding: 10px;
  background: radial - gradient(transparent 0px,transparent 4px,white 4px,white);
 /* 设置背景锯齿图形的大小 */
  background - size: 20px 20px;
 /* 设置锯齿图形的位置,10 是 20 的一半,所以最外边的透明圆刚好被切掉一半而像锯齿 */
  background - position: - 10px - 10px;
  - webkit - filter: drop - shadow(0px 0px 10px rgba(0,0,0,0.5)); }
  img{ width:280px; height:180px;}
 </style></head>
<body><div class = "myStamp" ><img src = "img/B358.jpg"/></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background：radial- gradient(transparent 0px，transparent 4px，white 4px，white)用于给盒子绘制透明圆，因为在默认情况下背景图形是重复摆放的，所以此行代码的显示效果如图 152-2 所示。如果禁止重复摆放，并且设置 background-size：20px 20px 和 background-position：－10px －10px，则可以看到 radial-gradient()方法产生的图形，如图 152-3 左上角所示的那个缺角的小正方形(20px 20px)。

此实例的源文件名是 myHtmlB365.html。

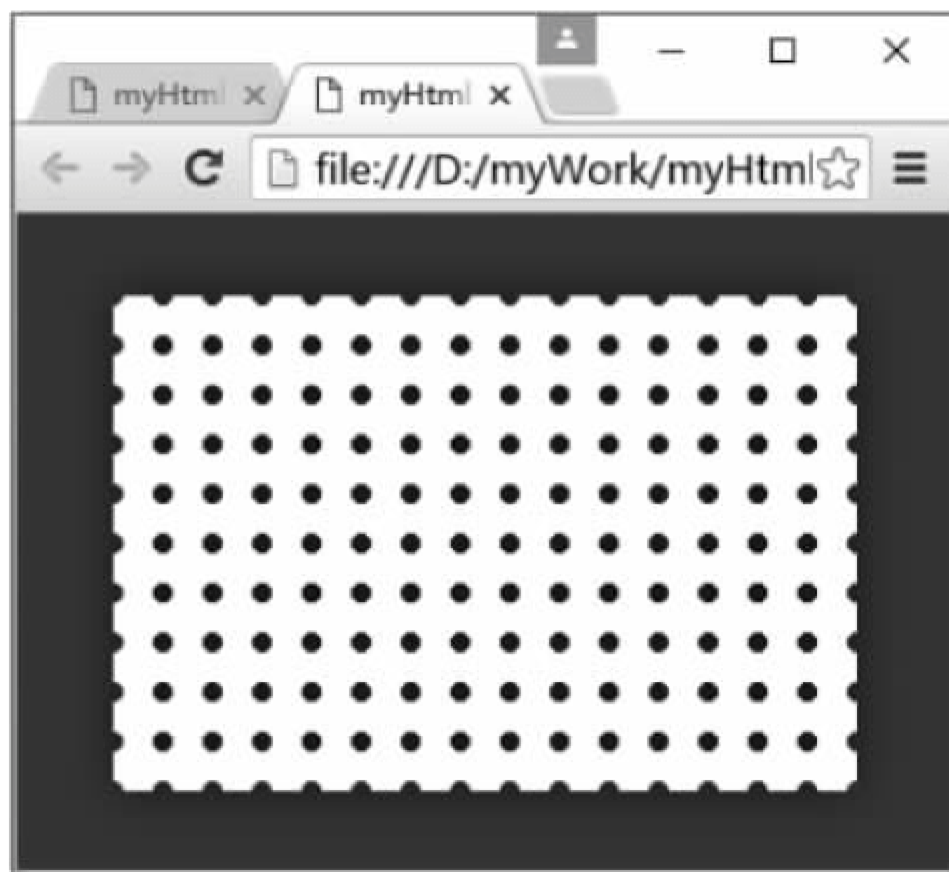


图 152-2

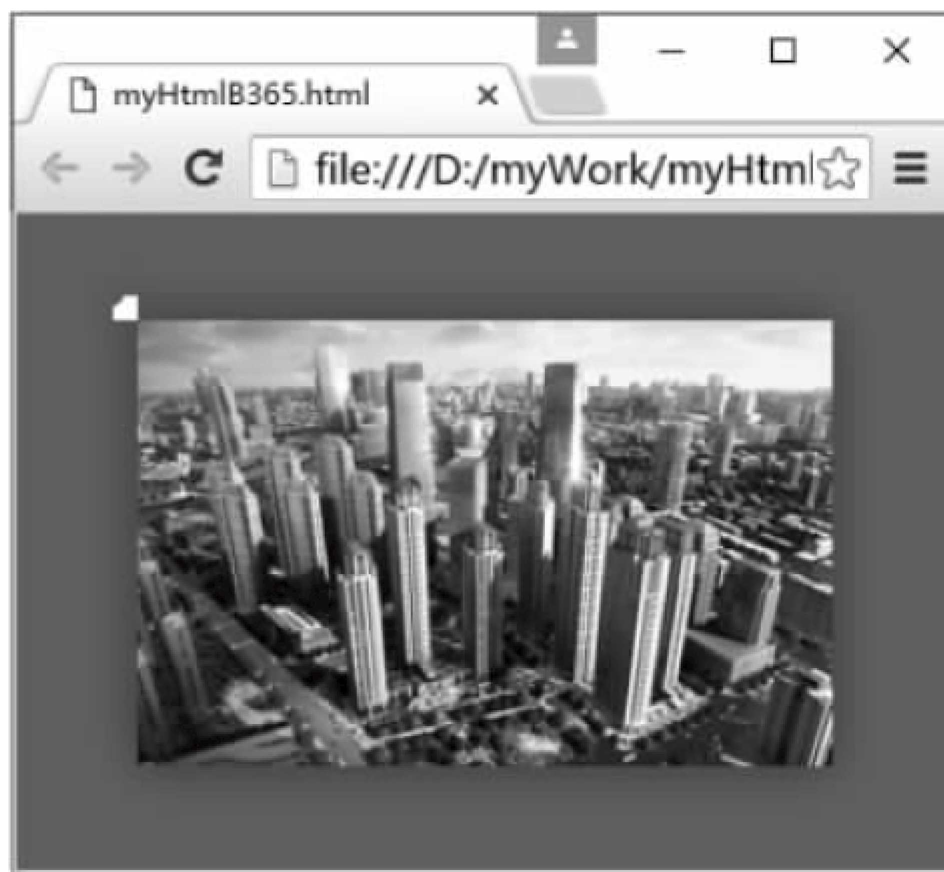


图 152-3

153 通过多级径向渐变创建美国队长的盾牌图案

此实例主要使用 radial-gradient()方法创建不同大小的圆环，从而使背景呈现美国队长的盾牌图案。当在 Google Chrome 浏览器中显示该页面时，美国队长的盾牌图案如图 153-1 所示。有关此实例的主要代码如下。



图 153-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myCaptainAmerica{ position: absolute; top: 50%;left: 50%;
     width: 200px;height: 200px;transform: translate(-50%, -50%);
     background:/*生成四射的渐变辉光*/
     linear-gradient(45deg, rgba(255,255,255,0) 35%, rgba(255,255,255,0.4) 50%, rgba(255,255,255,
0) 65%),
     linear-gradient(-45deg, rgba(255,255,255,0) 35%, rgba(255,255,255,0.4) 50%, rgba(255,255,
255,0) 65%),
     linear-gradient(to right, rgba(0,0,0,0) 35%, rgba(0,0,0,0.2) 50%, rgba(0,0,0,0) 65%),
```

```
        linear-gradient(to bottom, rgba(0,0,0,0) 35%, rgba(0,0,0,0.2) 50%, rgba(0,0,0,0) 65%),
        /*生成红白相间的圆环*/
        radial-gradient(ellipse at center, navy 20%, darkred 20%, darkred 35%, snow 35%, snow 55%,
    darkred 55%);
        border-radius: 50%; box-shadow: 0 3px 0 black;}
     /*在盾牌中心添加五角星*/
     .myCaptainAmerica::before{ content: '★'; position:absolute;
              top: 50%;left: 50%; transform: translate(-50%, -50%); z-index: 1;
              margin-left: -3px; width: 50px;height:50px; line-height: 47px;
              font-family: simsun,Tahoma,Helvetica,Arial,SimHei,sans-serif;
              border-radius: 50%; font-size: 55px; color: white;
              text-align:center; text-shadow: 2px 1px 2px black; }
    </style></head>
    <body><div class="myCaptainAmerica"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，radial-gradient(ellipse at center, navy 20%, darkred 20%, darkred 35%, snow 35%, snow 55%, darkred 55%)用于生成红白相间的圆环，这行代码也可以修改为 radial-gradient(navy 0%, navy 20%, darkred 20%, darkred 35%, snow 35%, snow 55%, darkred 55%, darkred 100%)。在 CSS3 中，radial-gradient()方法本质上是在径向生成颜色渐变，如果不同颜色在径向差值为 0，则这种径向颜色渐变就不会发生。因此，如果将代码修改为 radial-gradient(ellipse at center, navy 20%, darkred 20%, darkred 35%, snow 35%, snow 45%, darkred 55%)，即在最外层的红、白处有 55%－45%＝10%的差值，则会发生颜色渐变，如图 153-2 所示。如果这种差值很小，则可以实现去除毛刺的效果。例如，如果将代码修改为 radial-gradient(ellipse at center, navy 20%, darkred 20%, darkred 33%, snow 35%, snow 53%, darkred 55%)，则运行效果如图 153-3 所示，相比原图，圆环间的毛刺明显减少。

此实例的源文件名是 myHtmlB290.html。

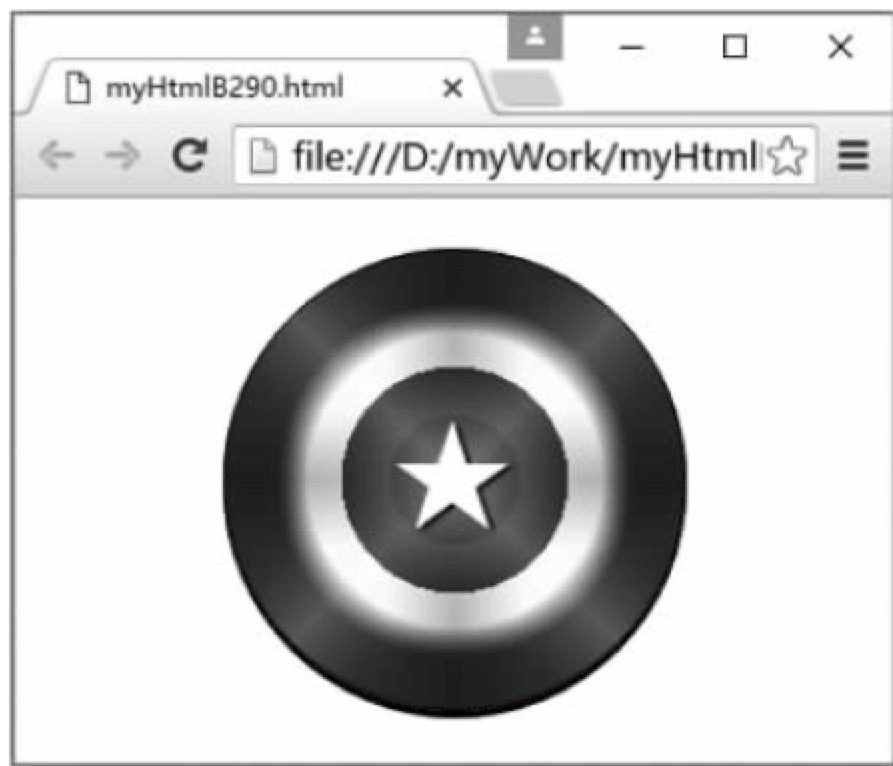


图 153-2

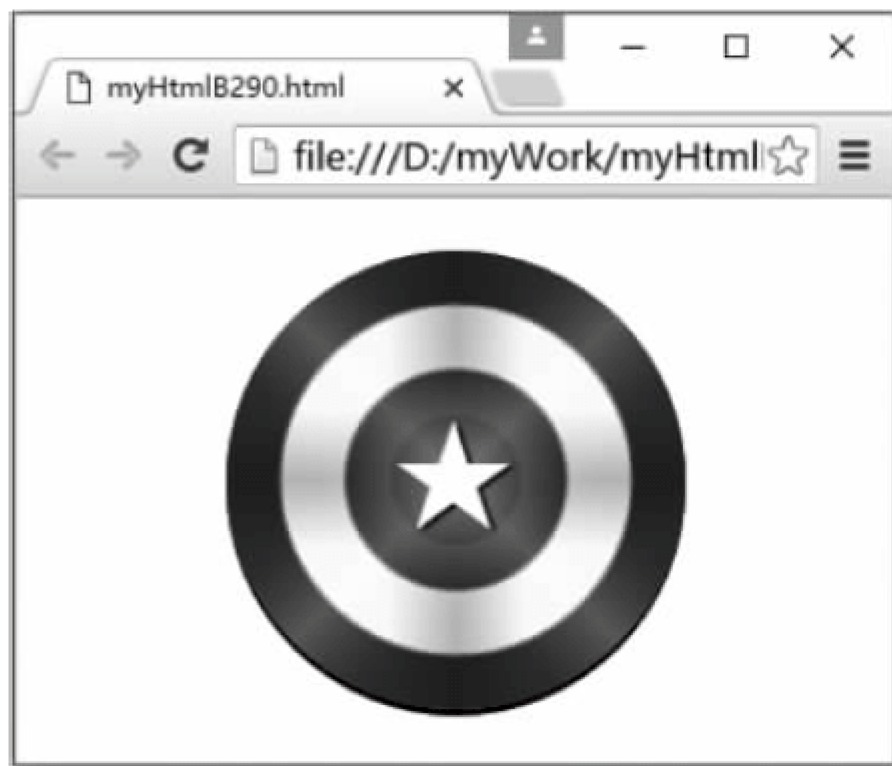


图 153-3

154 通过切分边框线所得的三角形创建五角星

此实例主要通过切分边框线获取三角形，然后进行放大处理，从而实现创建五角星的效果。当在 Google Chrome 浏览器中显示该页面时，创建的五角星如图 154-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myFivestar { position: absolute; top: 25%; right: 50%; width: 0; height: 0;
    transform: translate(-50%, -50%);display: block;
    /*以切分边框线的策略获得小三角形*/
    border-left: 3.04px solid transparent;border-right: 3.24px solid transparent;
    border-bottom: 10px solid #98DBE3;
    -webkit-filter: drop-shadow(1px .5px 1px #CCC);
    /*将小三角形放大8倍即得五角星上方的一个角*/
    transform: scale(8); }
  /*:before也以切分边框线的策略获得向左的小三角形*/
  .myFivestar:before { content: ""; position: absolute; top: 8.65px; left: -8.82px;
    width: 0; height: 0; color: #98DBE3; display: block;
    border-left: 12.5px solid transparent;border-right: 12.5px solid transparent;
    border-bottom: 9.08px solid #98DBE3;transform-origin: top center;
    transform: rotate(36deg); }
  /*:after仍以切分边框线的策略获得与:before反向的小三角形*/
  .myFivestar:after { content: ""; position: absolute;top: 8.65px; left: -15px;
    width: 0; height: 0;color: #98DBE3; display: block;
    border-left: 12.5px solid transparent; border-right: 12.5px solid transparent;
    border-bottom: 9.08px solid #98DBE3;transform-origin: top center;
    transform: rotate(-36deg); }
</style></head>
<body><div class="myFivestar"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-left 表示左边框线，border-right 表示右边框线，border-bottom 表示下(底部)边框线，这 3 条边框线没有长度(width：0;height：0;)，但有不同的宽度(如 12.5px、9.08px)。当左边框线与下(底部)边框线在左下角交汇时就会在对角线处剖分，因为 border-left 左边框线透明(transparent)，可见的只有下(底部)边框线的直角三角形；同时由于边框线没有长度(width：0;height：0;)，因此右边框线与下(底部)边框线在右下角交汇产生的直角三角形必定与左边框线与下(底部)边框线在左下角交汇产生的直角三角形连接在一起，即下面的代码产生的三角形(两个背靠背的直角三角形)放大 8 倍后如图 154-2 所示。



图 154-1



图 154-2

```
.myFivestar { position: absolute; top: 25%; right: 50%; display: block;
transform: translate(-50%, -50%);width: 0; height: 0;
/*以切分边框线的策略获得小三角形*/
border-left: 3.04px solid transparent;border-right: 3.24px solid transparent;
```

```
border - bottom: 10px solid #98DBE3;
-webkit-filter: drop-shadow(1px .5px 1px #CCC);
/*将小三角形放大8倍即得五角星上方的一个角*/
transform: scale(8);}
```

.myFivestar:before 和.myFivestar:after 产生三角形的原理与之类似。

此实例的源文件名是 myHtmlB288.html。

155 通过图形组合实现丝带缠绕展板的特效

此实例主要在 border-color 属性中合理地设置 transparent,从而实现折角丝带缠绕展板的特效。当在 Google Chrome 浏览器中显示该页面时,折角丝带缠绕展板的效果如图 155-1 的深粉色部分所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   /*设置盒子的基本样式*/
   .myBox { margin: 15px auto 0px auto;width: 400px; }
   /*设置文本区域的基本样式*/
   .myArea {clear: both; margin: 0px auto;width: 350px;background: cyan;
        position: relative; z-index: 90; -webkit-border-radius: 10px;
        -webkit-box-shadow: 0px 0px 8px rgba(0,0,0,0.3); }
   /*设置横丝带的基本样式*/
   .myRectangle { background: deeppink; height: 50px; width: 380px;
                  position: relative;left: -15px; top: 30px; float: left;
                  -webkit-box-shadow: 0px 0px 4px rgba(0,0,0,0.55);z-index: 100; }
   /*设置标题文本的基本样式*/
   .myRectangle p{ position: relative; top: -20px; font-size: 30px; color: black;
                  text-align: center; text-shadow: 1px 1px 2px rgba(0,0,0,0.2); }
   /*设置横丝带左下角的三角形的样式*/
   .myTriangle-l{border-color: transparent deeppink transparent transparent;
                  border-style:solid; border-width:15px;height:0px; width:0px;
                  position: relative; left: -30px; top: 65px; z-index: -1;}
   /*设置横丝带右下角的三角形的样式*/
   .myTriangle-r {border-color: transparent transparent transparent deeppink;
                  border-style:solid; border-width:15px; height:0px; width:0px;
                  position: relative;left: 350px; top: 35px; z-index: -1;}
   /*设置内容文本的基本样式*/
   .myContent { padding: 60px 25px 35px 25px;
               font-size:18px; padding-top:40px; padding-bottom: 20px;}
</style></head>
<body><div class="myBox"><div class="myArea">
  <div class="myRectangle" id="myRectangle"><p>重庆国际博览中心</p></div>
  <div class="myTriangle-l" id="myTriangle-l"></div>
  <div class="myTriangle-r" id="myTriangle-r"></div>
  <div class="myContent">重庆国际博览中心是一座集展览、会议、餐饮、住宿、演艺、赛事等多功能于一体的现代化智能场馆,位于重庆两江新区的核心 -- 悦来会展城,是西部最大的专业化场馆。重庆国际博览中心雄踞嘉陵江岸,依山傍水,公园环抱,古镇相伴,拥有城市、森林、自然浑然一体的优美环境,是国内独一无二的公园展馆、人文展馆、生态展馆。重庆国际博览中心总建筑面积达 60 万平方米,其中室内展览面积 20 万平方米。展馆共设 16 个展厅,南北各布置 8 个,为全国第二,西部第一。国博中心的屋顶呈蝴蝶形状。</div>
</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-color 属性用于设置 4 条边框线的颜色，此属性可设置一个元素的所有边框中可见部分的颜色，或者为 4 个边框线分别设置不同的颜色。border-color：red green blue pink 表示上边框是红色，右边框是绿色，下边框是蓝色，左边框是粉色；border-color：red green blue 表示上边框是红色，右边框和左边框是绿色，下边框是蓝色；border-color：dotted red green 表示上边框和下边框是红色，右边框和左边框是绿色。在此实例中，border-color：transparent deeppink transparent transparent 和 border-color：transparent transparent transparent deeppink 产生的边框线效果则是如图 155-2 展板的左、右两侧的两个紫色三角形（还需要同时设置 border-style：solid、border-width：15px、height：0px、width：0px；），实例通过下移标题栏的纵坐标遮挡两个紫色三角形的一半，因此产生了如图 155-1 所示丝带缠绕的视觉效果。

此实例的源文件名是 myHtmlB370.html。



图 155-1



图 155-2

156 在图像右上角创建倾斜 45°的梯形标签

此实例主要通过设置 overflow 和 transform 属性实现在图像右上角显示倾斜 45°的梯形标签。当在 Google Chrome 浏览器中显示该页面时，自定义标签“世界级图片”在图像右上角的显示效果如图 156-1 所示；如果鼠标指针悬浮在图片上，将显示另一张图片，自定义标签“世界级图片”在图像右上角仍然存在。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
   /＊设置图像框＊/
   .myLabel { text - align: center; margin: 10px auto; border - radius: 10px;
              width: 400px; height: 250px; background - size: 100 % 100 % ;
              - webkit - box - shadow: 5px 5px 8px rgba(0, 0, 0, 0.3);
              position: relative; background - image: url(img/B249A.jpg); }
```

```
    .myLabel:hover { background-image: url(img/B249B.jpg);cursor: hand; }
    /* 设置自定义标签框 */
    .myLabel.ribbon-myLabel-green { width: 85px; height: 88px;
            /* 裁剪掉溢出部分,自定义标签框则由矩形变成梯形 */
            overflow: hidden; position: absolute;
            /* 自定义标签框在右上角显示 */
            top: -1px; right: -1px; }
    .myLabel.ribbon-green { font: bold 15px Sans-Serif;text-align: center;
      text-shadow: rgba(255, 255, 255, 0.5) 0px 1px 0px;
      /* 自定义标签旋转 45° */
      -webkit-transform: rotate(45deg); position: relative; padding: 7px 0;
      left: -5px; top: 15px; width: 120px; color: black;
      /* 创建自定义标签渐变背景 */
      background-image: -webkit-linear-gradient(top, white, darkgreen);}
    .myLabel.ribbon-green:before, .ribbon-green:after { content: "";
      border-top: 3px solid darkgreen;border-left: 3px solid transparent;
      border-right: 3px solid transparent;position: absolute; bottom: -3px; }
    .myLabel.ribbon-green:before { left: 0;}        /* top 插入点 */
    .myLabel.ribbon-green:after { right: 0; }       /* right 插入点 */
  </style></head>
 <body><div class="myLabel">
  <div class="ribbon-myLabel-green">
   <div class="ribbon-green">世界级图片</div></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，overflow：hidden 用于裁剪掉自定义标签超出图像边缘的部分，因为自定义标签原本是普通的矩形，在裁剪了溢出部分之后则变成了梯形：-webkit-transform：rotate(45deg)表示将自定义标签旋转 45°。

此实例的源文件名是 myHtmlB249.html。

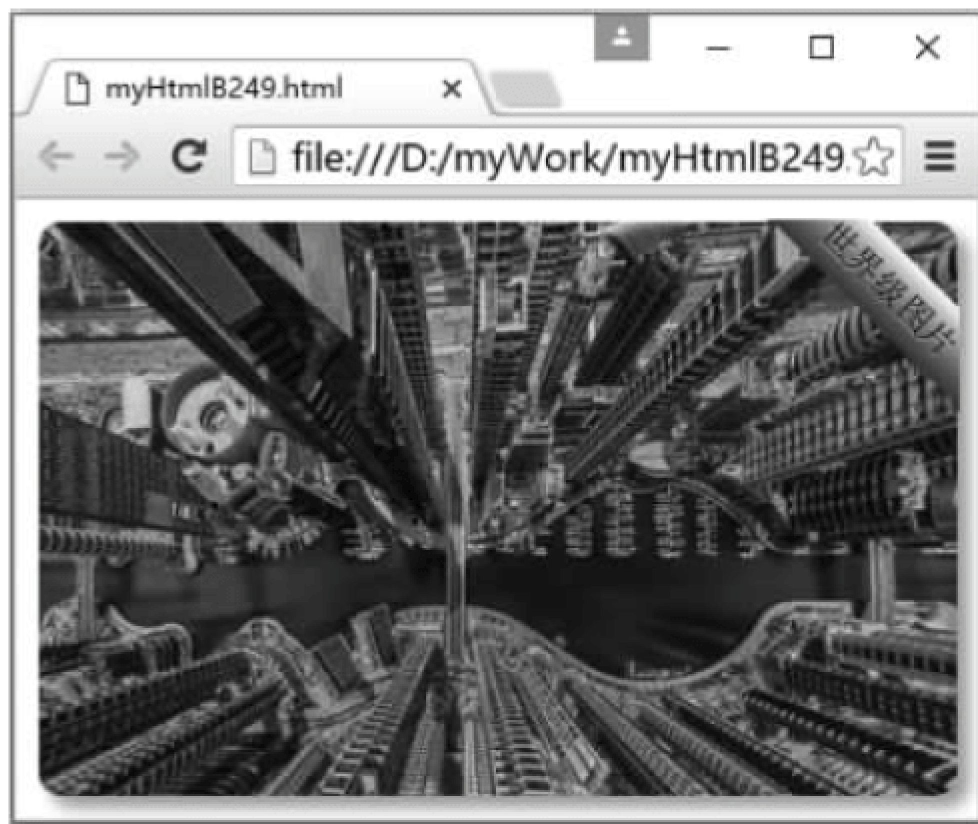


图　156-1

157　增加和移除图像左上角的自定义标签

此实例主要使用 jQuery 的 append()和 remove()方法，从而实现增加和移除图像左上角的自定义标签。当在 Google Chrome 浏览器中显示该页面时，单击“添加标签”按钮，则将在图像左上角添加自定义标签“望京球场”，如图 157-1 所示；单击“移除标签”按钮，则将移除图像左上角的自定义标签

“望京球场”，如图157-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#myBtnAdd").click(function(){//添加标签
    $("div.myBox").append("<p><span>望京球场</span></p>"); });
   $("#myBtnRemove").click(function(){//移除标签
    $("div.myBox p").remove(); });});
 </script>
 <style type="text/css">
  /*设置盒子的基本样式*/
  .myBox { margin-top:5px; position: relative; width:400px; height:250px;
          display: inline-block; overflow: hidden; border: 1px solid white;
          border-radius: 5px; -webkit-box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.3); }
          .myBox p { display: inline; }
  /*设置标签的样式*/
  .myBox p span { position: absolute; display: inline-block;text-align: center;
                 top:22px;background-image: -webkit-linear-gradient(top, white, blue);
                 color: black; width: 150px; padding: 3px 10px;
                 /*在左上角显示斜角标签*/
                 left: -50px; -webkit-transform: rotate(-45deg); }
  button{ width:195px; }
 </style></head>
<body><div align="center">
 <button id="myBtnAdd">添加标签</button>
 <button id="myBtnRemove">移除标签</button>
 <div class="myBox"><img src="img/B342.jpg"/></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div.myBox").append("<p>望京球场</p>")用于在图像的左上角添加标签p的自定义CSS样式；$("div.myBox p").remove()用于移除图像左上角的标签p。在jQuery中，append()方法用于在被选元素的结尾(仍然在内部)插入指定内容，该方法的语法格式如下。



图 157-1



图 157-2

```
$(selector).append(content)
```

其中，参数 content 规定要插入的内容(可包含 HTML 标签)。

remove()方法用于移除被选元素，包括所有文本和子结点，该方法不会把匹配的元素从 jQuery 对象中删除，因此可以在将来再使用这些匹配的元素；但除了这个元素本身得以保留之外，remove()方法不会保留元素的 jQuery 数据，其他的比如绑定的事件、附加的数据等都会被移除。remove()方法的语法格式如下。

```
$(selector).remove()
```

此实例的源文件名是 myHtmlB342.html。

158 以 6 面构建立方体并进行 3D 视觉旋转

此实例主要设置-webkit-perspective、-webkit-perspective-origin、-webkit- transform-style 等属性，从而实现使用 6 面构建立方体并进行 3D 视觉旋转。当在 Google Chrome 浏览器中显示该页面时，构建的立方体将在 10 秒内完成围绕 X 轴的 360°旋转，且永不停歇，如图 158-1 所示。有关此实例的主要代码如下。

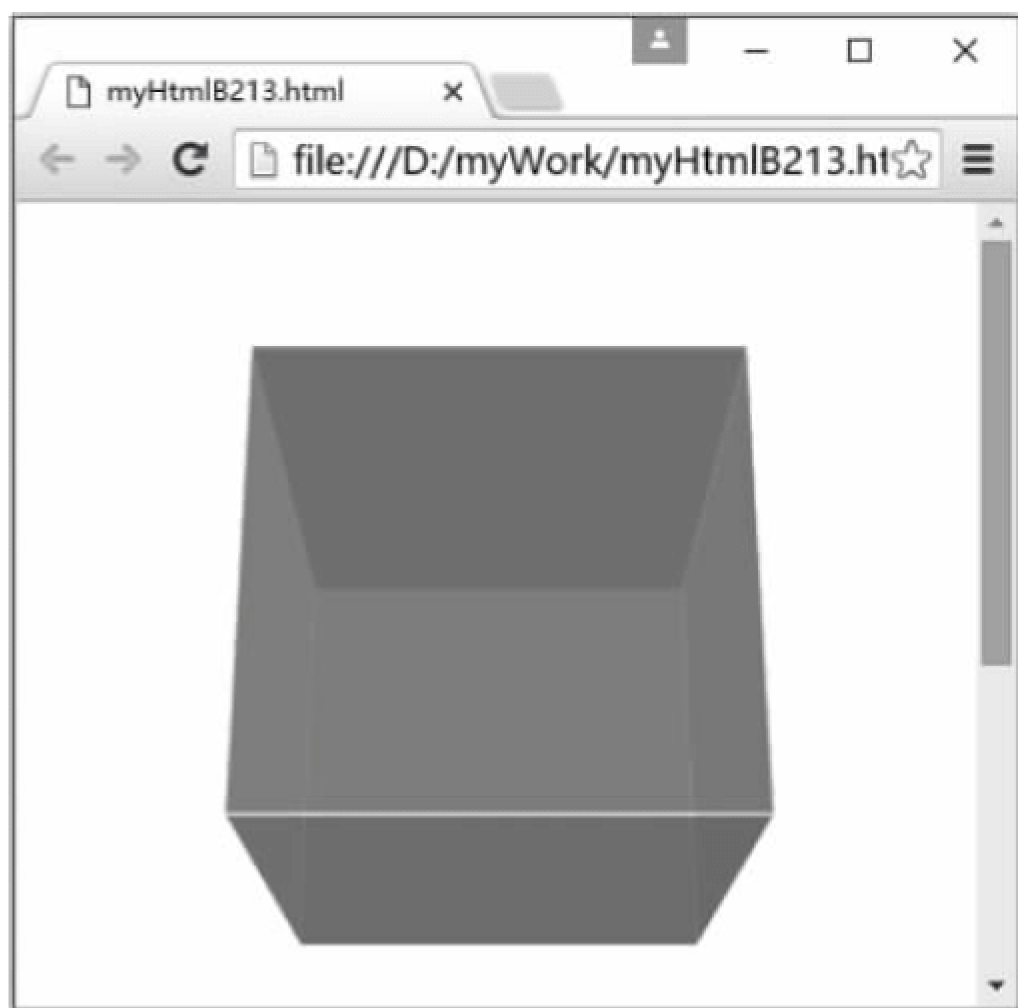


图 158-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   #outer { -webkit-perspective:600px;
            -webkit-perspective-origin: center center; }
   #group { width: 200px;height: 200px; position: relative;
            -webkit-transform-style: preserve-3d; margin: 200px auto;
            -webkit-animation: scroll 10s linear infinite; }
   @-webkit-keyframes scroll { 0% { -webkit-transform: rotateX(0deg) }
                               100% { -webkit-transform: rotateX(360deg) } }
   .myPlane { width: 200px; height: 200px; position: absolute;
              opacity: 0.9; border: 1px solid white; }
   #myPlane1 { background-color: #10A6CE; }
```

```
    #myPlane2 { background-color: #0073B3; -webkit-transform-origin: right;
                -webkit-transform: rotateY(-90deg); }
    #myPlane3 { background-color: #07BEEA; -webkit-transform-origin: left;
                -webkit-transform: rotateY(90deg); }
    #myPlane4 { background-color: #29B4F0; -webkit-transform-origin: top;
                -webkit-transform: rotateX(-90deg); }
    #myPlane5 { background-color: #6699CC; -webkit-transform-origin: bottom;
                -webkit-transform: rotateX(90deg); }
    #myPlane6 { background-color:navy; -webkit-transform: translateZ(-200px); }
  </style></head>
 <body><div id="outer"><div id="group">
   <div id="myPlane1" class="myPlane"></div>
   <div id="myPlane2" class="myPlane"></div>
   <div id="myPlane3" class="myPlane"></div>
   <div id="myPlane4" class="myPlane"></div>
   <div id="myPlane5" class="myPlane"></div>
   <div id="myPlane6" class="myPlane"></div></div></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-animation: scroll 10s linear infinite 表示在 10 秒内以线性方式完成 scroll 所指定的 360°旋转,且永不停歇;-webkit-transform: rotateX(360deg)表示将当前元素旋转 360°。当以 3D 视觉观察旋转时,通常还需要设置-webkit-perspective、-webkit-perspective-origin、-webkit-transform-style 等属性。-webkit-perspective 属性定义 3D 元素距视图的距离,以像素计算,该属性通常应该与 perspective-origin 属性同时使用。-webkit-perspective-origin 属性定义 3D 元素所基于的 X 轴和 Y 轴,该属性允许改变 3D 元素的底部位置。-webkit-perspective-origin 属性的语法格式如下。

```
-webkit-perspective-origin: x-axis y-axis;
```

其中,x-axis 定义该视图在 X 轴上的位置,默认值是 50%,可能的值有 left、center、right、length、%; y-axis 定义该视图在 Y 轴上的位置,默认值是 50%,可能的值有 top、center、bottom、length、%。

-webkit-transform-style 属性规定如何在 3D 空间中呈现被嵌套的元素,该属性的语法格式如下。

```
-webkit-transform-style: flat|preserve-3d;
```

其中,属性值 flat 表示子元素将不保留其 3D 位置;属性值 preserve-3d 表示子元素将保留其 3D 位置。

此实例的源文件名是 myHtmlB213.html。

159 改变透明度模拟飞碟在星空中穿梭的效果

此实例主要通过设置 opacity 属性控制飞碟图像在不同位置的透明度,从而实现模拟飞碟在星空中沿着椭圆形的轨道围绕星云穿梭时的时隐时显效果。当在 Google Chrome 浏览器中显示该页面时,6 个飞碟将围绕椭圆形的轨道飞行,如果飞碟飞到星云的背面位置,则以半透明效果显示,反之以正常效果显示,如图 159-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script type="text/javascript">
```

```
    window.onload = function() {
     var myUFO = new myRound("myBox", 120, 90, 300, 80, 230, 0.01);
     setInterval(function() {myUFO.roundMove() }, 20) }
    function myRound(id, w, h, x, y, r, dv, rh, ah) {
     if (ah == undefined) ah = 1;
     if (rh == undefined) rh = 10;
     var dv = dv * ah;                      //旋转速度
     var pi = 3.1415926575;var d = pi/2;
     var pd = Math.asin(w/2/r);
     var smove = true; var myImgs = [];
     var objectId = id;
     var o = document.getElementById(objectId);
     o.style.position = "relative";
     var myArray = o.getElementsByTagName("img");
     var pn = myArray.length;               //图片数量
     var ed = pi * 2/pn;
     for (n = 0; n < myArray.length; n++) {      //当鼠标指针离开 UFO 时继续飞行
      myArray[n].onmouseout = function() { smove = true; }
      myArray[n].onmouseover = function() {  //当鼠标指针悬浮在 UFO 上停止飞行
       smove = false; }
      myArray[n].style.position = "absolute";
      myImgs.push(myArray[n]);
    }
    this.roundMove = function() {
    for (n = 0; n <= pn - 1; n++) {
     var o = myImgs[n];
     var ta = Math.sin(d + ed * n), strFilter;
     if (ta < 0)
      o.style.left = Math.cos(d + ed * n - pd) * r + x + "px";
     else
      o.style.left = Math.cos(d + ed * n + pd) * r + x + "px";
     o.style.top = ta * rh + rh + y + "px";
     var zoom = Math.abs(Math.sin((d + ed * n)/2 + pi/4)) * 0.5 + 0.5;
     o.style.width = Math.abs(Math.cos(d + ed * n + pd) - Math.cos(d + ed * n - pd)) * zoom * r + "px";
     o.style.height = zoom * h + "px";
     if (ta < 0) {
      ta = (ta + 1) * 80 + 20; o.style.zIndex = 0;
     }else { ta = 100; o.style.zIndex = 1 }
     o.style.opacity = ta/100;              //改变飞碟在不同位置的透明度
    }
    if (smove)
      d = d + dv; } }
  </script>
  <! -- 设置星空背景 -->
  <style type = "text/css">
    div { width: 600px; height: 300px;background:url("img/a119B.jpg");
          border - radius: 10px; }
  </style></head>
  <body>
  <! -- 设置绕轨道飞行的飞碟图像 -->
  <div id = "myBox">
      <img src = "img/A119.png"/><img src = "img/A119.png"/>
      <img src = "img/A119.png"/><img src = "img/A119.png"/>
      <img src = "img/A119.png"/><img src = "img/A119.png"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，o.style.opacity＝ta/100 用于设置飞碟图像的透明度。在 CSS 中，opacity 属性用于设置不透明级别，该属性值的范围是 0.0(完全透明)～1.0(完全不透明)，因此在实例代码中使用了 100 这个除数。

此实例的源文件名是 myHtmlB320.html。

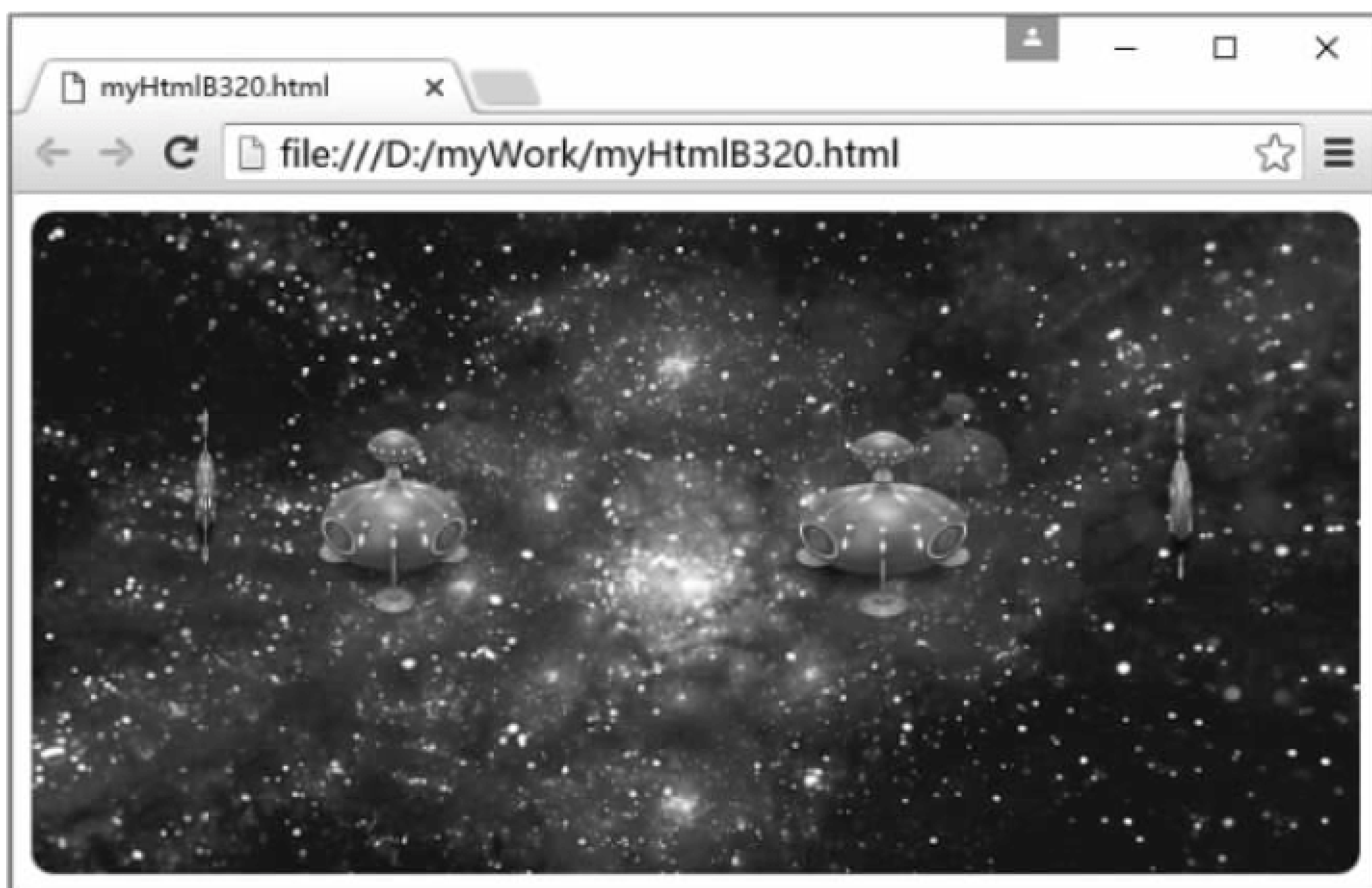


图 159-1

160 根据鼠标指针的轨迹确定如何旋转和平移图像

此实例主要在 mousemove 和 mouseleave 事件中使用 translate()、rotateX()、rotateY()等方法，从而实现根据鼠标指针的滑动轨迹确定如何旋转和平移图像。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针在图像上从左上角向右下角滑动，则图像在旋转和平移之后的效果如图 160-1 所示；如果鼠标指针在图像上从右上角向左下角滑动，则图像在旋转和平移之后的效果如图 160-2 所示。有关此实例的主要代码如下。

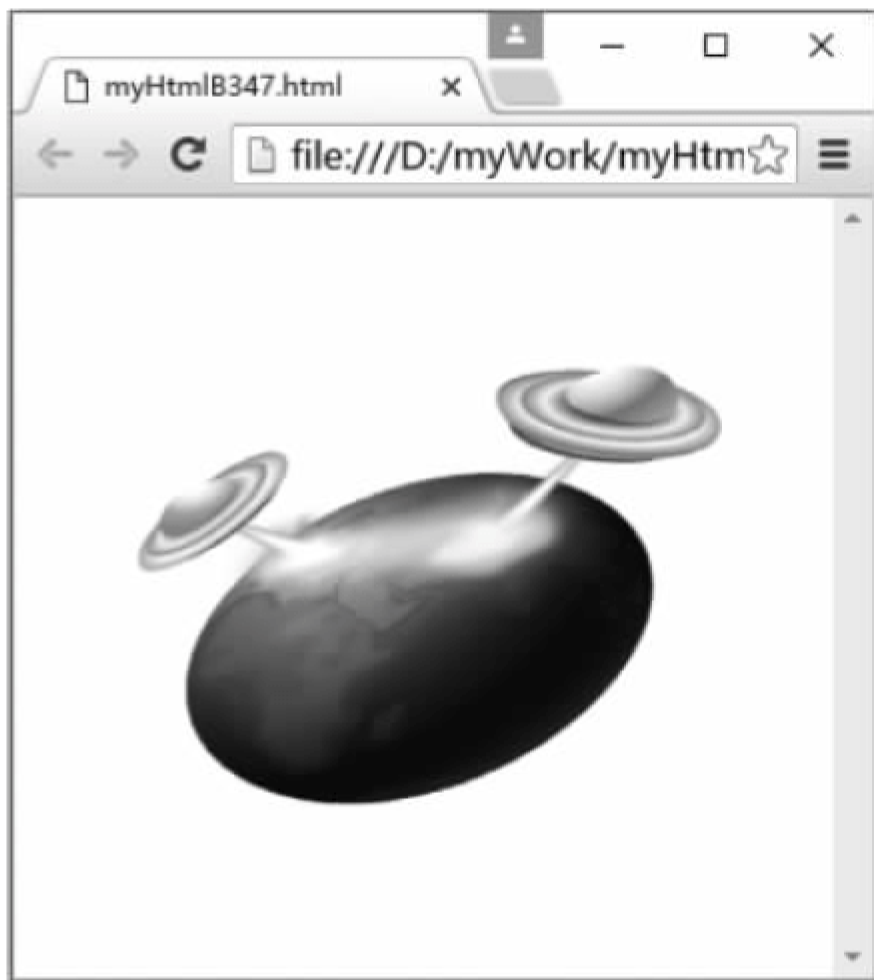


图 160-1

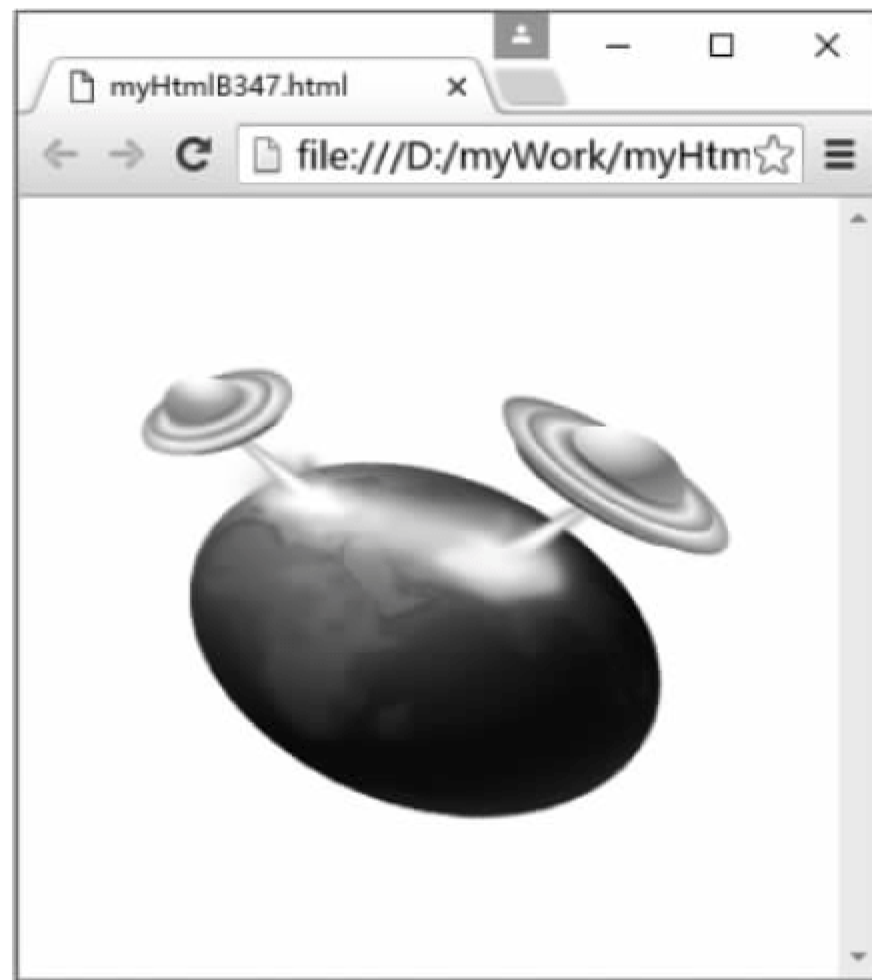


图 160-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  function animate3dElement(element,distance,rotationMultiple,startX,startY, animate,setCssStyles){
   var myOffset,myTop,myLeft,myWidth,myHeight;myOffset=$(element).offset();
   myTop=myOffset.top;myLeft=myOffset.left;
   myWidth=$(element).innerWidth();myHeight=$(element).innerHeight();
   animate(element, startX, startY, rotationMultiple, distance, setCssStyles);
    $(element).mousemove(function(event){          //响应鼠标指针移动事件
     var pctX=((event.pageX-myLeft)/myWidth)-0.5+startX;
     var pctY=((event.pageY-myTop)/myHeight)-0.5+startY;
     animate(this, pctX, pctY, rotationMultiple, distance, setCssStyles);
   });
    $(element).mouseleave(function(){              //响应鼠标指针离开事件
     animate(this, startX, startY, rotationMultiple, distance, setCssStyles);
  });}
  //旋转和平移图像的回调函数
  function animate (element, pctX, pctY, rotationMultiple, distance, callback){
   var rotateX=pctY*rotationMultiple*-180;
   var rotateY=pctX*rotationMultiple*180;
   var translateX=pctX*distance; var translateY=pctY*distance;
    $(element).children().each(function(index, element){
     callback(element, index, translateX, translateY, rotateX, rotateY);
   });};
  function setCssStyle(element, index, translateX, translateY, rotateX, rotateY, distance){ //设置图像
的旋转和平移
   var x=translateX*index;var y=translateY*index;
    $(element).css('-webkit-transform', 'translate('+ x +'px ,'+ y +'px) rotateX('+ rotateX +'deg)
rotateY('+ rotateY +'deg)');
  };
  $(function(){
   animate3dElement('.myAnimate',1000,0.5,0,0,animate,setCssStyle);
    $(window).resize(function(){
     animate3dElement('.myAnimate',1000,0.5,0,0,animate,setCssStyle); });});
 </script>
 <style type="text/css">
  .myBox { width: 100%; padding-bottom: 65%; }    /* 设置盒子的基本样式 */
  /* 设置图像的基本样式 */
  .myBox img { width: 80%;left: 10%; top: 10%; position: absolute; }
 </style></head>
<body><div align="center"><div class="myBox myAnimate">
 <img src="img/B302.png" /></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(element).css('-webkit-transform', 'translate('+ x +'px ,'+ y +'px) rotateX('+ rotateX +'deg) rotateY('+ rotateY +'deg)')用于根据指定的参数对图像进行旋转和平移。在 CSS 中，translate(x,y)方法用于根据参数指定的 x、y 坐标平移元素，rotateX(angle)方法用于根据参数围绕 X 轴旋转指定的角度，rotateY(angle)方法用于根据参数围绕 Y 轴旋转指定的角度。event.pageX 和 event.pageY 用于在鼠标指针滑动时获取其坐标位置。

此实例的源文件名是 myHtmlB347.html。

161　使用负数参数获取图像的水平和垂直镜像

此实例主要通过在 scaleX()和 scaleY()方法中使用负数参数获取图像的水平和垂直镜像。当在 Google Chrome 浏览器中显示该页面时将显示原始的图像；单击“显示水平镜像”按钮，该图像的水平镜像效果如图 161-1 所示；单击“显示垂直镜像”按钮，该图像的垂直镜像效果如图 161-2 所示。有关此实例的主要代码如下。



图　161-1



图　161-2

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myHorizontalMirror").click(function() {          //显示水平镜像
    $("img").css("-webkit-transform", "scaleX(-1)"); });
   $("#myVerticalMirror").click(function() {            //显示垂直镜像
    $("img").css("-webkit-transform", "scaleY(-1)"); });
   $("#myInitial").click(function() {                   //显示原始图像
    $("img").css("-webkit-transform", "initial"); });});
 </script>
 <style type = "text/css">
   * { margin: 0; padding: 0; }
   img{ width:350px; height:350px; }
   button{ width:120px; margin-right: 5px; margin-top: 15px; }
 </style></head>
<body><div align = "center">
 <button id = "myHorizontalMirror">显示水平镜像</button>
 <button id = "myVerticalMirror">显示垂直镜像</button>
 <button id = "myInitial">显示原始图像</button>
 <p><img src = "img/B207A.png"/></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css("-webkit-transform","scaleX(-1)")用于显示图像的水平镜像，$("img").css("-webkit-transform","scaleY(-1)")用于显示图像的垂直镜像。在 CSS3 中，scaleX(x)用于通过设置 X 轴的值来定义水平方向的缩放转换，但是如果参数为负数，将以水平镜像方式完成缩放；scaleY(y)用于通过设置 Y 轴的值来定义垂直方向的缩放转换，但是如果参数为负数，将以垂直镜像方式完成缩放。

此实例的源文件名是 myHtmlB356.html。

162　以纯 CSS 使用两幅图像实现星级评分特效

此实例主要通过使用两幅图像以纯粹的 CSS 代码实现了星级评分特效。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第 3 颗星上，则会显示一个浮动框提示信息，例如“上将”，如图 162-1 所示，单击则会保存选择结果。有关此实例的主要代码如下。

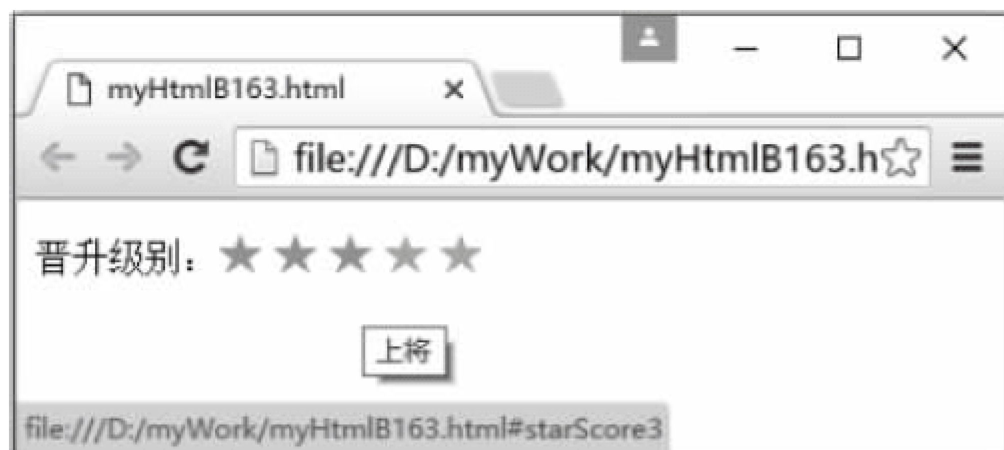


图　162-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .star_bg { top:5px; width: 120px; height: 20px; overflow: hidden;
           background: url(img/B163.png) repeat-x;position: relative; }
  .star { height: 100%; width: 24px;line-height: 6em;
         position: absolute; z-index:6;}
  .star:hover { background: url(img/B163.png) repeat-x 0 -20px!important;
               left: 0; z-index: 4; }
  .star_1 { left: 0; }
  .star_2 { left: 24px; }
  .star_3 { left: 48px; }
  .star_4 { left: 72px; }
  .star_5 { left: 96px; }
  .star_1:hover { width: 24px; }
  .star_2:hover { width: 48px; }
  .star_3:hover { width: 72px; }
  .star_4:hover { width: 96px; }
  .star_5:hover { width: 120px; }
  label {display: block;height: 100%; width: 100%;cursor: pointer;}
  .score { position: absolute; clip: rect(0 0 0 0); }
  .score:checked+.star { background: url(img/B163.png) repeat-x 0 -20px;
                        left: 0; z-index: 1;}
  .score_1:checked ~ .star_1 { width: 24px; }
  .score_2:checked ~ .star_2 { width: 48px; }
  .score_3:checked ~ .star_3 { width: 72px; }
  .score_4:checked ~ .star_4 { width: 96px; }
  .score_5:checked ~ .star_5 { width: 120px; }
```

```
    .star_bg:hover .star { background-image: none; }
    p { float: left; text-align: center;line-height: 2px;}
  </style></head>
<body><p>晋升级别: <div id="starBg" class="star_bg">
  <input type="radio" id="starScore1" class="score score_1" value="1" name="score">
  <a href="#starScore1" class="star star_1" title="少将"><label for="starScore1">少将</label></a>
  <input type="radio" id="starScore2" class="score score_2" value="2" name="score">
  <a href="#starScore2" class="star star_2" title="中将"><label for="starScore2">中将</label></a>
  <input type="radio" id="starScore3" class="score score_3" value="3" name="score">
  <a href="#starScore3" class="star star_3" title="上将"><label for="starScore3">上将</label></a>
  <input type="radio" id="starScore4" class="score score_4" value="4" name="score">
  <a href="#starScore4" class="star star_4" title="大将"><label for="starScore4">大将</label></a>
  <input type="radio" id="starScore5" class="score score_5" value="5" name="score">
  <a href="#starScore5" class="star star_5" title="元帅"><label for="starScore5">元帅</label></a>
</div></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，z-index 属性用于设置元素的堆叠顺序，拥有更高堆叠顺序的元素总会处于堆叠顺序较低的元素的前面。在此实例中，.star 的 z-index 属性值 6 大于.star:hover 的 z-index 属性值 4，因此它总出现在其前面，具体值无关紧要，只比较相对大小。clip 属性用于剪裁绝对定位元素，其语法格式如下。

```
clip:auto | <shape>
<shape>:rect(<number>|auto <number>|auto <number>|auto <number>|auto)
```

其中，auto 表示对象无剪切。rect(<number>|auto <number>|auto <number>|auto <number>|auto)表示依据上-右-下-左的顺序提供自对象左上角为(0,0)坐标计算的 4 个偏移数值，其中任一数值都可用 auto 替换，即此边不剪切。上-左方位的裁剪为从 0 开始剪裁直到设定值，即上-左方位的 auto 值等同于 0；右-下方位的裁剪为从设定值开始剪裁直到最右边和最下边，即右-下方位的 auto 值为盒子的实际宽度和高度，例如 clip:rect(auto 50px 20px auto)表示上边不剪切，右边从左起第 50 个像素开始剪切直到最右边，下边从上起第 20 个像素开始剪切直到最底部，左边不剪切。在此实例中，clip: rect(0 0 0 0)实际上是全部裁剪。

此实例的源文件名是 myHtmlB163.html。

163 使用盒子阴影模拟半透明的遮罩层特效

此实例主要通过设置 box-shadow 的阴影尺寸超大实现在图像的周围产生半透明的遮罩层。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针未悬浮在图像上，则显示的 4 幅图像如图 163-1 所示；如果鼠标指针悬浮在第一幅图像上，则放大该图像并在图像的周围产生半透明的遮罩层，如图 163-2 所示。当然，如果鼠标指标悬浮在其他图像上，也能放大对应的图像并在该图像的周围产生半透明的遮罩层。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   * { margin: 0; padding: 0; }
  /* 盒子水平居中 */
  div{ margin: 50px auto; width:410px; height:155px;}
  img { border-radius: 5px; float: left; height: 150px; width: 100px;
     cursor:pointer; border: 1px solid lightgreen;
     /* 使用过渡动画放大图像 */
```

```
        transition:transform 0.2s ease;}
    /* 鼠标指针悬浮时产生遮罩层并放大图像 */
    img:hover{ box-shadow:0 0 0 1920px rgba(0,0,0,0.8);transform: scale(1.5); }
  </style></head>
<body><div><img src="img/B268A.jpg"><img src="img/B268B.jpg">
      <img src="img/B268C.jpg"><img src="img/B268D.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow:0 0 0 1920px rgba(0,0,0,0.8)用于在鼠标指针悬浮于图像上时在其周围产生一个尺寸为1920px的半透明阴影，因为1920px这个尺寸足以覆盖目前的普通显示器屏幕，所以具有与遮罩层相同的效果。在CSS3中，box-shadow属性用于向盒子添加一个或多个阴影，该属性由逗号分隔阴影列表，每个阴影由2～4个长度值、可选的颜色值以及可选的inset关键字来规定，省略长度的值是0。

此实例的源文件名是myHtmlB268.html。

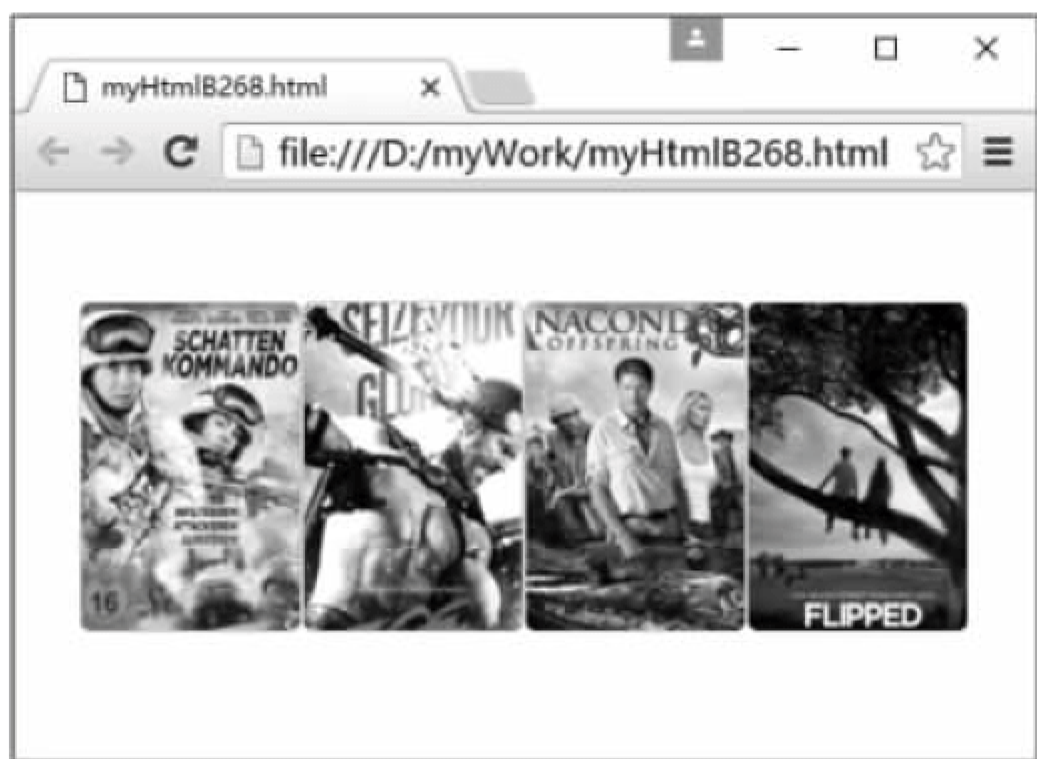


图 163-1

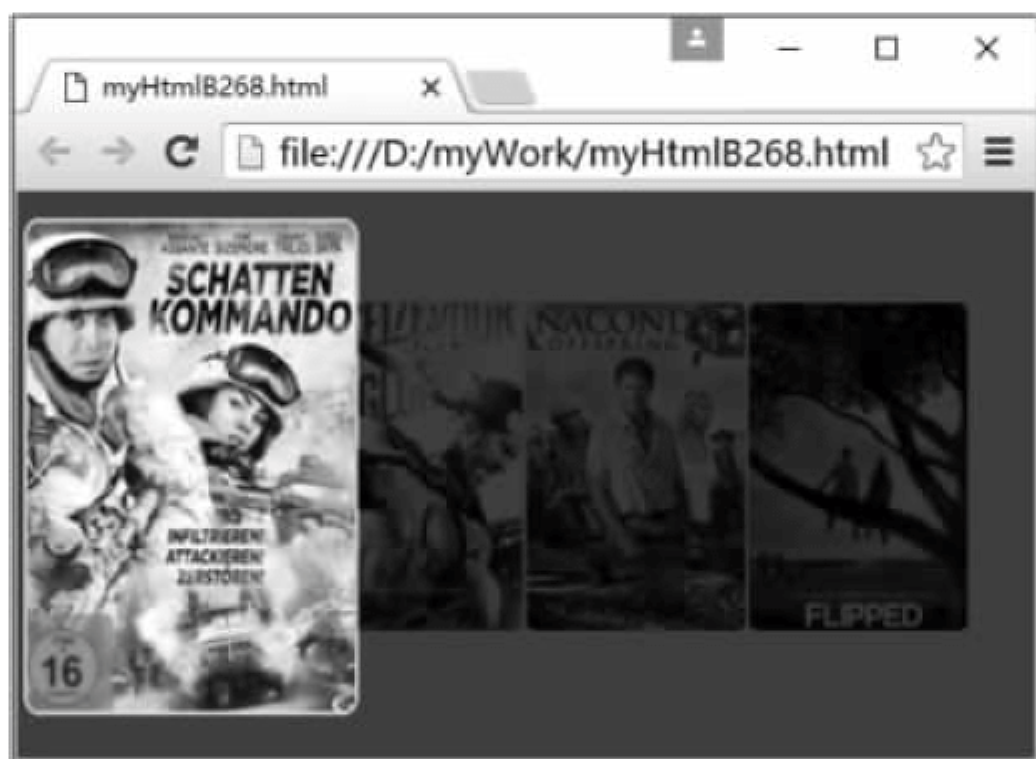


图 163-2

164 通过使用线性渐变实现纸张卷角效果

此实例主要通过在线性渐变方法中规定颜色(透明)范围创建背景图像，从而实现纸张卷角的特效。当在Google Chrome浏览器中显示该页面时，纸张卷角的效果如图164-1所示。有关此实例的主要代码如下。



图 164-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  body { background: lightgrey;}
  .myCorner { position: absolute; top: 50%; left: 50%; border-radius:0.5em;
             transform: translate(-50%, -50%);width: 350px; padding: 40px;
             /*通过透明裁剪右上角*/
             background: linear-gradient(-150deg, transparent 1.5em, green 0);}
  /*通过 transparent 和 box-shadow 生成卷角阴影*/
  .myCorner:before { content: ''; position: absolute;top: 0; right: 0;
        /*通过透明裁剪卷角矩形的另一半*/
        background: linear-gradient(to left bottom, transparent 50%, rgba(0, 0, 0, 0.1) 9%, rgba(0,
0, 0, 0.9) 99%);
        background-repeat: no-repeat; width: 1.73em; height: 3em;
        transform: translateY(-1.3em) rotate(-30deg);
        transform-origin: bottom right;border-bottom-left-radius: inherit;
        box-shadow: -0.2em 0.2em 0.3em -0.1em rgba(0, 0, 0, 0.15);}
</style></head>
<body><div class=" myCorner">1945年,第二次世界大战接近尾声,作为邪恶轴心重要成员的日本,其嚣张态
势已成强弩之末。是年,决定战局走向的冲绳岛战役拉开序幕,成千上万斗志昂扬的美国大兵被派往冲绳,等待
他们的则是敌军重兵防守、凶险异常的钢锯岭。</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background: linear-gradient(−150deg, transparent 1.5em, green 0)表示透明色从右上角开始至 1.5em,然后背景的其余部分用绿色填充,它的作用就是用透明色裁剪右上角,如图 164-2 所示; background: linear-gradient(to left bottom, transparent 50%, rgba(0, 0, 0, 0.1) 9%, rgba(0, 0, 0, 0.9) 99%)表示完全透明色从右上角开始至对角线(50%),然后背景的其余部分用不同的透明度叠加在下面的元素上,这个矩形在被添加阴影之后旋转 30°放在裁角的位置上就形成了卷角,如图 164-3 所示。

此实例的源文件名是 myHtmlB278.html。



图 164-2



图 164-3

165 使用模糊和灰度滤镜处理未选中图像

此实例主要使用 not 选择器筛选 hover 选择器之外的图像,从而对未选中的图像使用模糊滤镜和灰度滤镜进行处理。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在左边的图像上,将以清晰方式显示该图像,其他图像则以模糊和灰度效果显示,如图 165-1 所示;如果鼠标指针悬浮在中间或右边的图像上,将以清晰方式显示该图像,其他图像则以模糊和灰度效果显示。有关此实例的主要代码如下。

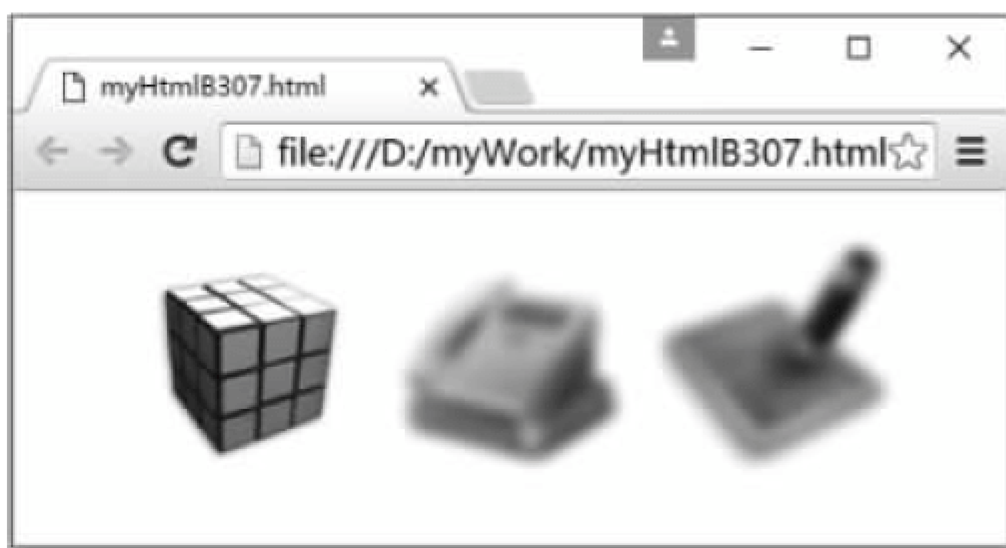


图 165-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  img { width: 100px; height: 100px; margin: 10px; }
  ul, li { list-style: none;}
  ul li { float: left; }
  /*表示未选中的图像,使用灰度滤镜和模糊滤镜处理图像*/
  img:not(:hover) { -webkit-filter: grayscale(1) blur(3px); }
 </style></head>
<body>
<ul><li><img src="img/B065.jpg"/></li><li><img src="img/B066A.png"/></li>
    <li><img src="img/B209A.png"/></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，img:not(:hover){ }表示选择不在 hover 状态的 img 元素。在 CSS3 中，hover 选择器用于选择鼠标指针浮动在上面的元素；not 选择器则用于匹配非指定元素/选择器的每个元素。

此实例的源文件名是 myHtmlB307.html。

166　在弹出提示框时模糊页面中的其他部分

此实例主要对 body 使用模糊滤镜，从而实现在弹出提示框时模糊显示页面的其他部分。当在 Google Chrome 浏览器中显示该页面时，单击“开园时间”按钮，将正常弹出一个提示框，但是页面的其他部分(例如按钮、图像、背景等)变得模糊，如图 166-1 所示，在关闭提示框之后页面恢复正常显示；单击“门票价格”按钮，将实现类似的功能。有关此实例的主要代码如下。

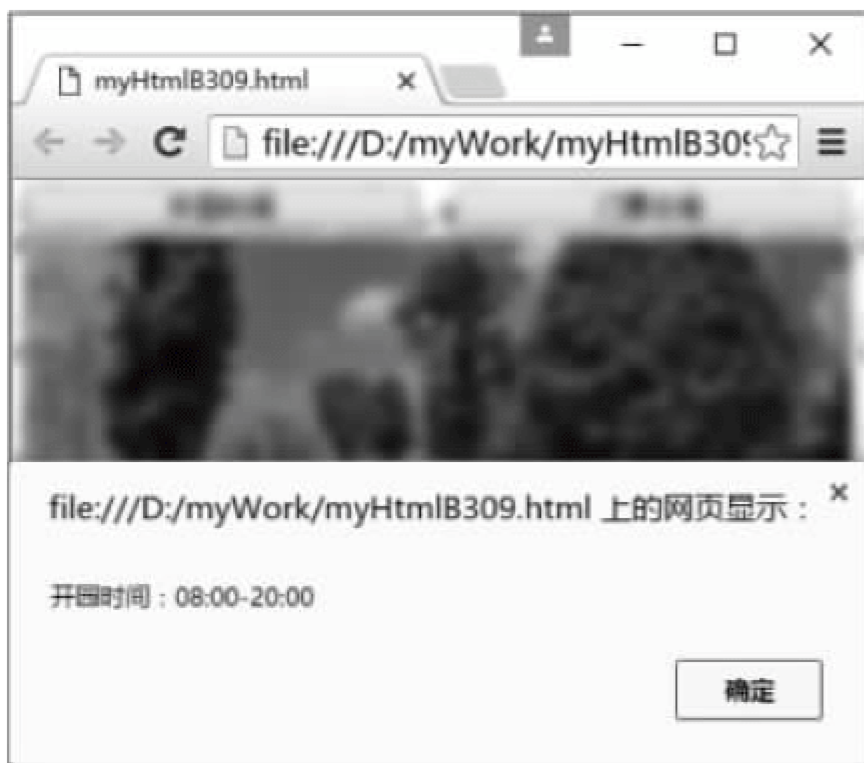


图 166-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnTime").click(function() {//开园时间
    $(document.body).css('-webkit-filter', 'blur(3px)');
    alert('开园时间: 08:00-20:00');
    $(document.body).css('-webkit-filter', 'blur(0px)');
   });
   $("#myBtnPrice").click(function() {//门票价格
    $(document.body).css('-webkit-filter', 'blur(3px)');
    alert("",'门票价格: 20元/人');
    $(document.body).css('-webkit-filter', 'blur(0px)'); });});
 </script>
 <style type="text/css">
  *{ margin: 0; }
  .myBody{ height:100vh; width:100vw; background:url(img/B055A.png); }
  button { width: 190px; margin: 3px 5px; }
  img{ width:400px; height:250px; border-radius: 5px; margin: 1px 5px;}
 </style></head>
<body>
<div class="myBody"><p><button id="myBtnTime">开园时间</button>
                      <button id="myBtnPrice">门票价格</button></p>
    <img src="img/B179A.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(document.body).css('-webkit-filter','blur(3px)')表示对document.body中的所有元素模糊3px，3px表示模糊半径。

此实例的源文件名是myHtmlB309.html。

167 使用任意颜色设置png图像的轮廓颜色

此实例主要通过drop-shadow()实现使用任意颜色设置png图像的文字轮廓颜色。当在Google Chrome浏览器中显示该页面时，艺术字"生日快乐"的颜色是粉色的，单击"请选择颜色："右边的颜色选择器按钮，在弹出的"颜色"对话框中任意选择一种颜色，例如"青色"，然后单击"设置新的颜色"按钮，则将以青色显示艺术字"生日快乐"，如图167-1所示。有关此实例的主要代码如下。



图 167-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
    $("#myBtnSet").click(function(){//设置新的颜色
     $("img").css("-webkit-filter",
         "drop-shadow(384px 0 0px "+$("input[type=color]").val()+")");
  });});
 </script>
 <style type="text/css">
  body{ background: black;}
  img{ width:384px; height:250px; position:relative;
       /* 坐标为负值主要是为了隐藏原始图像 */
       left:-384px; top:-30px;border-right:384px solid transparent;
       /* 用户看到的图像则是下面这行代码产生的阴影 */
       -webkit-filter:drop-shadow(384px 0 0px pink);}
  .myBox{ width:390px; height:450px; overflow:hidden; }
  button{ margin-top:10px; width:200px; }
  span{ color: snow; }
 </style></head>
<body><div align="center">
 <div><span>请选择颜色: </span><input type="color">
       <button id="myBtnSet">设置新的颜色</button></div>
<div class="myBox"><img src="img/B360.png" width="384" height="250"/></div></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-filter:drop- shadow(384px 0 0px pink)表示使用滤镜为元素(img 标签)产生一个粉色阴影,384px 表示阴影在 X 轴上的偏移距离,由于元素(img 标签)已经设置 left:-384px 和 width:384px,所以用户看到的只是粉色阴影,原始图像不可见。

此实例的源文件名是 myHtmlB360.html。

168　模拟不用的扑克牌始终插在最后的效果

此实例主要通过控制元素(扑克牌)的 z-index 属性值保持最小来模拟不用的扑克牌始终插在一叠的最后。当在 Google Chrome 浏览器中显示该页面时,4 张扑克牌叠在一起,使用鼠标单击最上面的扑克牌(红桃 8),则该扑克牌将向右滑动,然后向左滑动插在此叠扑克牌的最后面,如图 168-1 所示,此时最上面的扑克牌变成大王;再次使用鼠标单击最上面的扑克牌(大王),则该扑克牌将向右滑动,然后向左滑动插在此叠扑克牌的最后面,如图 168-2 所示,此时最上面的扑克牌变成红桃 K,以此类推。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
    $(function() {
    var z=-1;
     $("p").click(function() {         //响应使用鼠标单击扑克牌
      $(this).animate({left: "120px"}, 1000, function() {
       //保持当前的扑克牌的 z-index 属性值始终是最小的
        $(this).css("z-index", z--)
     }).animate({left: "10px"}, 1000); }) });
```

```
</script>
 <style type = "text/css">
   p { width: 108px; height: 145px; z - index: 0;
       position: absolute; left: 10px; border - radius: 15px; }
   p img { width: 108px;height: 145px; border - radius: 10px; }
   body { background - color: lightgray; }
   div{ position: absolute; margin: 10px 20 % ; }
 </style></head>
<body><div><p><img src = "img/B265A.jpg"/></p><p><img src = "img/B265B.jpg"/></p>
     <p><img src = "img/B265C.jpg"/></p><p><img src = "img/B265D.jpg"/></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(this).css("z-index", z--)用于在当前扑克牌向右滑动之后立即设置其 z-index 属性为最小值，因此它向左滑动时始终在最后面。在 CSS 中，z-index 属性用于设置元素的堆叠顺序，拥有更高堆叠顺序的元素总是处于堆叠顺序较低的元素的前面，即该属性是设置一个定位元素沿 Z 轴的位置，Z 轴定义为垂直延伸到显示区的轴；如果为正数，则离用户更近，为负数表示离用户更远。

此实例的源文件名是 myHtmlB321.html。

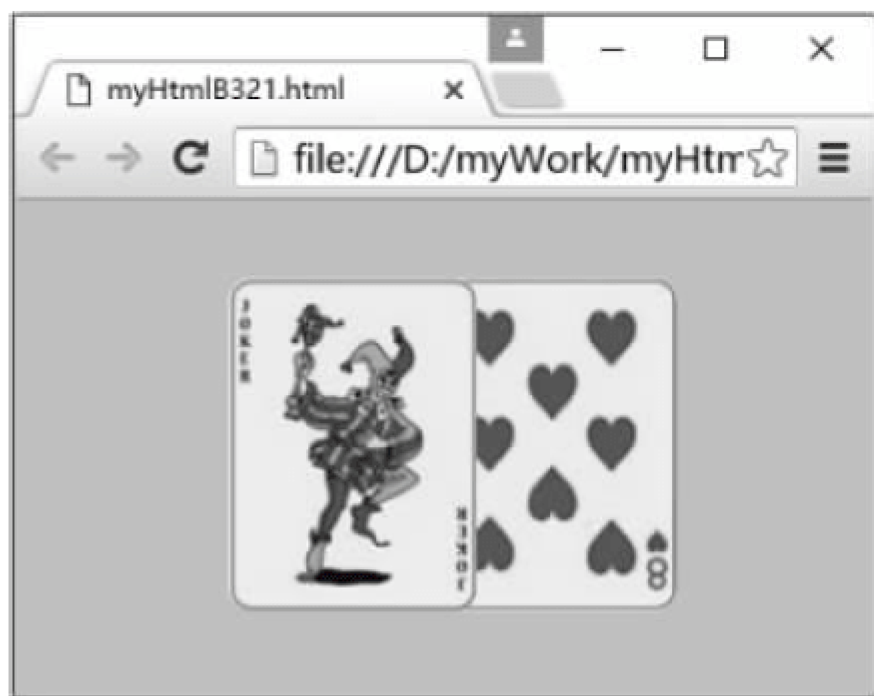


图 168-1



图 168-2

169 将元素的属性值作为图像的标题显示

此实例主要在 content 属性中通过“attr(属性名)”这种形式来获取(超链接)元素的属性值，从而实现将该属性值作为图像的标题来显示。当在 Google Chrome 浏览器中显示该页面时将通过 attr(title)获取两个超链接的 title 属性值，并将该属性值作为图书的书名来显示，如图 169-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  a:after{content:attr(title);display: block; text - align: center;
   margin - top:1px; width:200px;font - size: 18px;font - weight: bold;
   color: black; margin - left: 80px; font - family: 华文隶书;}
  a{text - decoration:none;margin - left: 80px;}
  img{ width: 200px;height: 200px;margin: 5px;}
 </style></head>
<body><p><a href = "http://item.jd.com/11144230.html" title = "算法导论(原书第 3 版)">
<img src = "img/B027A.jpg"/></a></p>
```

```
<p><a href = "http://item.jd.com/10951037.html" title = "JavaScript 高级程序设计" > < img src = "img/
B027B.jpg"/></a></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，在页面上用(图书封面)图像来显示超链接，并在超链接中指定 title 属性值；在图像下面显示的书名是在 CSS 样式中通过“content:attr(title)”这种形式获取的超链接信息，然后就可以按照常用的方式设置书名的字体、大小、颜色等属性。

此实例的源文件名是 myHtmlB027.html。



图　169-1

170　使用 AlloyImage 对图像进行柔化处理

此实例主要通过使用腾讯公司的开源插件 AlloyImage 对图像进行柔化特效处理。当在浏览器中显示该页面时将显示未经处理的图像，单击“柔化图像”按钮，则图像将变得模糊起来，如图 170-1 所示；单击“撤销柔化”按钮，图像将恢复原样。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/alloyimage.js"></script><script language = "javascript">
  $(document).ready(function() {
   var myImage = $("#myJpg").get(0);
   var luoImage = AlloyImage(myImage);
   $("#myBtnSoften").click(function() {          //柔化图像
    AlloyImage(myImage).ps("soften").replace(myImage);
   });
```

```
    $("#myBtnUndo").click(function() {          //撤销柔化
     luoImage.replace(myImage); });});
  </script></head>
 <body><img id="myJpg" src="img/a031.jpg" />
 <p><input type="button" value="柔化图像" id="myBtnSoften" style="width:180px">
 <input type="button" value="撤销柔化" id="myBtnUndo" style="width:195px"></p>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage). ps("soften"). replace(myImage)中的 myImage 是将被处理的图像，soften 表示对图像进行柔化处理。

此实例的源文件名是 myHtmlA031. html。



图 170-1

171 使用 AlloyImage 对图像进行黑白处理

此实例主要通过使用腾讯公司的开源插件 AlloyImage 将彩色图像转换为黑白图像。当在浏览器中显示该页面时将显示未经处理的图像，单击"转换为黑白图像"按钮，图像将以黑白效果显示，如图 171-1 所示；单击"撤销转换"按钮，图像将恢复原样。有关此实例的主要代码如下。



图 171-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script><script language="javascript">
   $(document).ready(function() {
    var myImage=$("#myJpg").get(0);
    var luoImage=AlloyImage(myImage);
    $("#myBtnStrongGray").click(function() {               //转换为黑白图像
     AlloyImage(myImage).ps("strongGray").replace(myImage);
    });
    $("#myBtnUndo").click(function() {                     //撤销转换
     luoImage.replace(myImage);
 });});
 </script></head>
<body><img id="myJpg" src="img/a031.jpg" />
<p><input type="button" value="转换为黑白图像" id="myBtnStrongGray" style="width:180px"/>
   <input type="button" value="撤销转换" id="myBtnUndo" style="width:195px"/>
</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("strongGray").replace(myImage)中的myImage是将被处理的图像，strongGray表示对图像进行增强型灰化处理，即黑白效果。

此实例的源文件名是myHtmlA032.html。

172　使用AlloyImage对图像进行素描处理

此实例主要通过使用腾讯公司的开源插件AlloyImage对图像进行素描特效处理。当在浏览器中显示该页面时将显示未经处理的图像，单击“对图像进行素描处理”按钮，图像将以铅笔素描效果显示，如图172-1所示；单击“撤销处理”按钮，图像将恢复原样。有关此实例的主要代码如下。



图　172-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script><script language="javascript">
```

```
   $ (document).ready(function() {
    var myImage = $ ("#myJpg").get(0);
    var luoImage = AlloyImage(myImage);
     $ ("#myBtnSketch").click(function(){        //对图像进行素描处理
      AlloyImage(myImage).ps("sketch").replace(myImage);
    });
     $ ("#myBtnUndo").click(function() {         //撤销处理
      luoImage.replace(myImage);
   });});
 </script></head>
<body><img id="myJpg" src="img/a033.jpg" />
<p><input type="button" value="对图像进行素描处理" id="myBtnSketch" style="width:190px">
    <input type="button" value="撤销处理" id="myBtnUndo" style="width:190px">
</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("sketch").replace(myImage)中的 myImage 是将被处理的图像，sketch 表示对图像进行素描特效处理。

此实例的源文件名是 myHtmlA033.html。

173　使用 AlloyImage 给图像添加 LOMO 特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加 LOMO 特效。当在浏览器中显示该页面时将显示未经处理的图像，单击"给图像添加 LOMO 特效"按钮，图像的亮度、对比度、噪音等参数都将改变，如图 173-1 所示；单击"撤销特效"按钮，图像将恢复原样。有关此实例的主要代码如下。



图　173-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script>
 <script language="javascript">
   $ (document).ready(function() {
```

```
    var myImage = $("#myJpg").get(0);
    var luoImage = AlloyImage(myImage);
    $("#myBtnLomo").click(function() {           //给图像添加 LOMO 特效
     for(var i = 0;i<3;i++){
      AlloyImage(myImage).ps("lomo").replace(myImage);
    } });
    $("#myBtnUndo").click(function() {           //撤销特效
     luoImage.replace(myImage);
   });});
  </script></head>
 <body>
 <img id = "myJpg" src = "img/a034.jpg" />
 <p><input type = "button" value = "给图像添加 LOMO 特效" id = "myBtnLomo" style = "width:190px">
   <input type = "button" value = "撤销特效" id = "myBtnUndo" style = "width:190px">
 </p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("lomo").replace(myImage)中的 myImage 是将被处理的图像，lomo 表示对图像进行 LOMO 特效处理。

此实例的源文件名是 myHtmlA034.html。

174 使用 AlloyImage 给图像添加暖秋特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加暖秋特效。当在浏览器中显示该页面时将显示未经处理的图像，单击“给图像添加暖秋特效”按钮，图像的部分像素值将发生改变，如图 174-1 所示；单击“撤销特效”按钮，图像将恢复原样。有关此实例的主要代码如下。

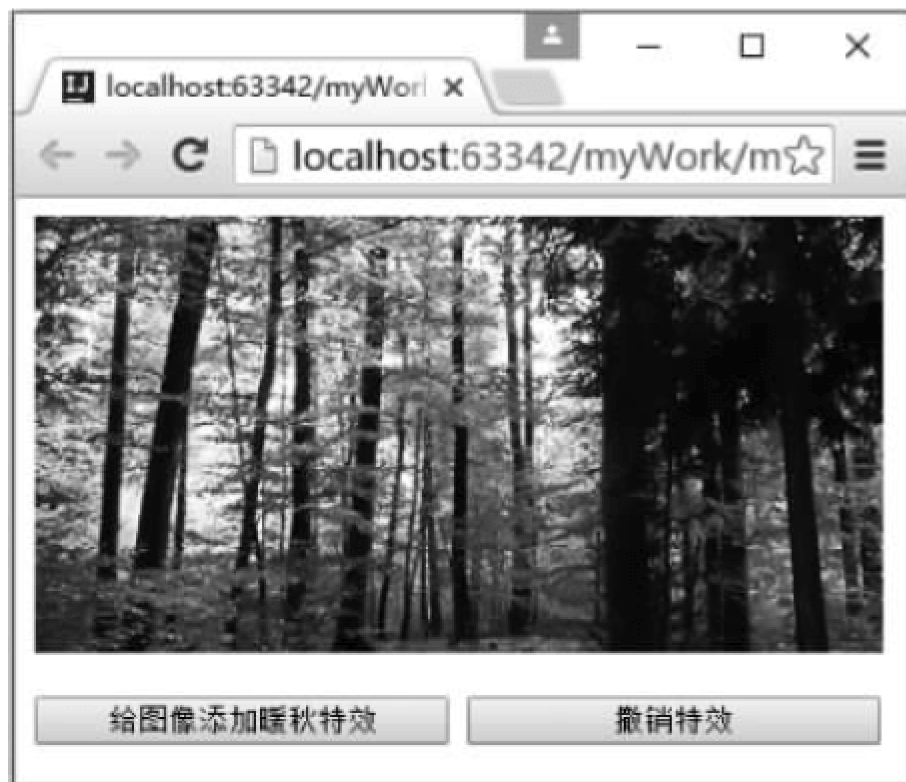


图 174-1

```
<!doctype html><html><head><meta charset = UTF-8>
 <script src = "js/jquery-3.1.1.min.js"></script>
 <script src = "js/alloyimage.js"></script>
 <script language = "javascript">
   $(document).ready(function() {
    var myImage = $("#myJpg").get(0);
    var luoImage = AlloyImage(myImage);
```

```
      $("#myBtnAutumn").click(function() {          //给图像添加暖秋特效
       for(var i=0;i<1;i++){
        AlloyImage(myImage).ps("warmAutumn").replace(myImage);
      }});
      $("#myBtnUndo").click(function() {            //撤销特效
       luoImage.replace(myImage);
      });});
  </script></head>
 <body><img id="myJpg" src="img/a035.jpg" />
 <p><input type="button" value="给图像添加暖秋特效" id="myBtnAutumn" style="width:180px">
    <input type="button" value="撤销特效" id="myBtnUndo" style="width:180px"></p>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("warmAutumn").replace(myImage)中的myImage是将被处理的图像，warmAutumn表示给图像添加暖秋特效。

此实例的源文件名是myHtmlA035.html。

175　使用AlloyImage给图像添加粗糙特效

此实例主要通过使用腾讯公司的开源插件AlloyImage给图像添加粗糙特效。当在浏览器中显示该页面时将显示未经处理的图像，单击“给图像添加粗糙特效”按钮，图像将发生类似于马赛克的改变，如图175-1所示；单击“撤销特效”按钮，图像将恢复原样。有关此实例的主要代码如下。



图　175-1

```
 <!doctype html><html><head><meta charset=UTF-8>
  <script src="js/jquery-3.1.1.min.js"></script>
  <script src="js/alloyimage.js"></script><script language="javascript">
    $(document).ready(function() {
     var myImage=$("#myJpg").get(0);
     var luoImage=AlloyImage(myImage);
      $("#myBtnRough").click(function() {           //给图像添加粗糙特效
       for(var i=0;i<1;i++){
        AlloyImage(myImage).ps("rough").replace(myImage);
```

```
        }
      });
      $ ("#myBtnUndo").click(function() {          //撤销特效
       luoImage.replace(myImage);
   }); });
  </script></head>
 <body><img id="myJpg" src="img/a036.jpg" />
 <p><input type="button" value="给图像添加粗糙特效" id="myBtnRough" style="width:180px">
  <input type="button" value="撤销特效" id="myBtnUndo" style="width:180px">
 </p>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage). ps("rough"). replace(myImage)中的 myImage 是将被处理的图像，rough 表示给图像添加粗糙特效。

此实例的源文件名是 myHtmlA036.html。

176 使用 AlloyImage 给图像添加紫色特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加紫色特效。当在浏览器中显示该页面时将显示未经处理的图像，单击"给图像添加紫色特效"按钮，图像添加紫色特效之后的效果如图 176-1 所示；单击"撤销特效"按钮，图像将恢复原样。有关此实例的主要代码如下。

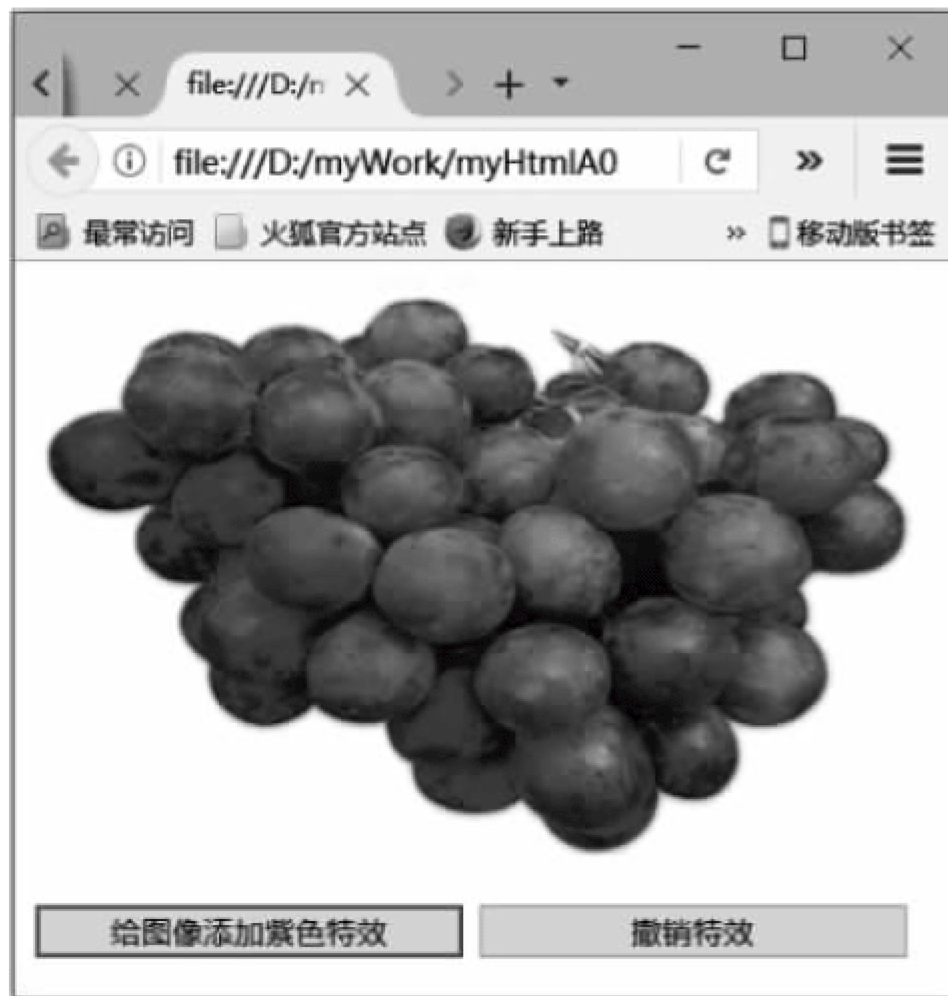


图 176-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script><script language="javascript">
   $ (document).ready(function() {
    var myImage = $ ("#myJpg").get(0);
    var luoImage = AlloyImage(myImage);
     $ ("#myBtnPurple").click(function() {          //给图像添加紫色特效
       AlloyImage(myImage).ps("purpleStyle").replace(myImage);
```

```
    });
    $("#myBtnUndo").click(function() {              //撤销特效
     luoImage.replace(myImage);
   });});
  </script></head>
 <body><img id="myJpg" src="img/a037.jpg" />
 <p><input type="button" value="给图像添加紫色特效" id="myBtnPurple" style="width:188px"><input
 type="button" value="撤销特效" id="myBtnUndo" style="width:188px"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("purpleStyle").replace(myImage)中的 myImage 是将被处理的图像，purpleStyle 表示给图像添加紫色特效。

此实例的源文件名是 myHtmlA037.html。

177　使用 AlloyImage 给图像添加复古特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加复古特效。当在浏览器中显示该页面时将显示未经处理的图像，单击“给图像添加复古特效”按钮，图像经过复古处理之后的效果如图 177-1 所示；单击“撤销特效”按钮，图像将恢复原样。有关此实例的主要代码如下。



图　177-1

```
 <!doctype html><html><head><meta charset=UTF-8>
  <script src="js/jquery-3.1.1.min.js"></script>
  <script src="js/alloyimage.js"></script><script language="javascript">
   $(document).ready(function() {
    var myImage=$("#myJpg").get(0);
    var luoImage=AlloyImage(myImage);
    $("#myBtnVintage").click(function() {           //给图像添加复古特效
     for(var i=0;i<1;i++){
      AlloyImage(myImage).ps("vintage").replace(myImage);
    } });
    $("#myBtnUndo").click(function() {              //撤销特效
     luoImage.replace(myImage);
   });});
```

```
</script></head>
<body><img id="myJpg" src="img/a038.jpg" />
<p><input type="button" value="给图像添加复古特效" id="myBtnVintage" style="width:195px">
<input type="button" value="撤销特效" id="myBtnUndo" style="width:190px"></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("vintage").replace(myImage)中的 myImage 是将被处理的图像，vintage 表示对图像进行复古特效处理。

此实例的源文件名是 myHtmlA038.html。

178 使用 AlloyImage 给图像添加木雕特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加木雕特效。当在浏览器中显示该页面时将显示未经处理的图像，单击“给图像添加木雕特效”按钮，图像经过木雕处理之后的效果如图 178-1 所示；单击“撤销特效”按钮，图像将恢复原样。有关此实例的主要代码如下。

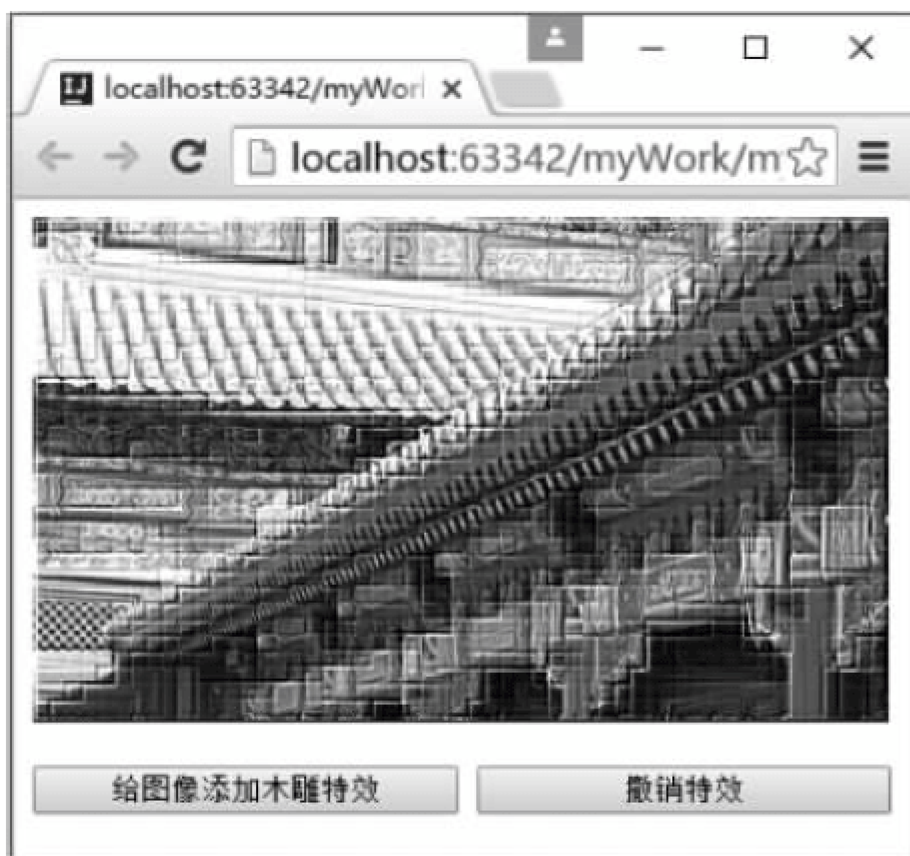


图 178-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script><script language="javascript">
  $(document).ready(function() {
   var myImage=$("#myJpg").get(0);
   var luoImage=AlloyImage(myImage);
   $("#myBtnCarve").click(function() {          //给图像添加木雕特效
    for(var i=0;i<1;i++){
     AlloyImage(myImage).ps("carveStyle").replace(myImage);
   } });
   $("#myBtnUndo").click(function() {           //撤销特效
    luoImage.replace(myImage);
  });});
 </script></head>
<body><img id="myJpg" src="img/a039.jpg" />
<p><input type="button" value="给图像添加木雕特效" id="myBtnCarve" style="width:185px"><input
type="button" value="撤销特效" id="myBtnUndo" style="width:180px"></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("carveStyle").replace(myImage)中的 myImage 是将被处理的图像，carveStyle 表示对图像进行木雕特效处理。

此实例的源文件名是 myHtmlA039.html。

179 使用 AlloyImage 给图像添加美肤特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加美肤特效。当在浏览器中显示该页面时将显示未经处理的图像，单击“给图像添加美肤特效”按钮，图像经过美肤处理之后的效果如图 179-1 所示；单击“撤销特效”按钮，图像将恢复原样。有关此实例的主要代码如下。



图 179-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/alloyimage.js"></script>
 <script language="javascript">
  $(document).ready(function() {
   var myImage=$("#myJpg").get(0);
   var luoImage=AlloyImage(myImage);
   $("#myBtnFace").click(function() {//给图像添加美肤特效
    for(var i=0;i<1;i++){
     AlloyImage(myImage).ps("softenFace").replace(myImage);
   } });
   $("#myBtnUndo").click(function() {//撤销特效
    luoImage.replace(myImage);
  });});
 </script></head>
<body><img id="myJpg" src="img/a040.jpg" />
<p><input type="button" value="给图像添加美肤特效" id="myBtnFace" style="width:185px"><input
type="button" value="撤销特效" id="myBtnUndo" style="width:185px"></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("softenFace").replace(myImage)中的 myImage 是将被处理的图像，softenFace 表示对图像进行美肤特效处理。

此实例的源文件名是 myHtmlA040.html。

180　使用 AlloyImage 给图像添加亮白特效

此实例主要通过使用腾讯公司的开源插件 AlloyImage 给图像添加亮白特效。当在浏览器中显示该页面时将显示未经处理的图像，单击"给图像添加亮白特效"按钮，图像经过亮白处理之后的效果如图 180-1 所示；单击"撤销特效"按钮，图像将恢复原样。有关此实例的主要代码如下。



图　180-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/alloyimage.js"></script><script language = "javascript">
  $ (document).ready(function() {
   var myImage = $ ("#myJpg").get(0);
   var luoImage = AlloyImage(myImage);
   $ ("#myBtnStrong").click(function() {//给图像添加亮白特效
    for(var i = 0;i<1;i++){
      AlloyImage(myImage).ps("strongEnhancement").replace(myImage);
   } });
   $ ("#myBtnUndo").click(function() {//撤销特效
     luoImage.replace(myImage);
  });});
 </script></head>
<body><img id = "myJpg" src = "img/a041.jpg" />
<p><input type = "button" value = "给图像添加亮白特效" id = "myBtnStrong" style = "width:190px"><input
type = "button" value = "撤销特效" id = "myBtnUndo" style = "width:185px"></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，AlloyImage(myImage).ps("strongEnhancement").replace(myImage)中的 myImage 是将被处理的图像，strongEnhancement 表示对图像进行亮白特效处理。

此实例的源文件名是 myHtmlA041.html。

181 在 SVG 中使用高斯模糊滤镜处理图像

此实例主要通过在 SVG 中使用高斯模糊滤镜实现以模糊的效果显示图像。当在 Google Chrome 浏览器中显示该页面时，单击“正常显示图像”按钮，将在下面正常显示图像；单击“使用高斯模糊滤镜显示图像”按钮，将在下面以高斯模糊效果显示图像，如图 181-1 所示。有关此实例的主要代码如下。



图 181-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnBlur").click(function() {//使用高斯模糊滤镜显示图像
    $("svg defs filter feGaussianBlur").attr("stdDeviation", 3);
   });
   $("#myBtnNormal").click(function() {//正常显示图像
    $("svg defs filter feGaussianBlur").attr("stdDeviation", 0);
  });});
 </script>
 <style type="text/css">
  svg { margin-top: 20px; position: absolute;left: calc(50% - 200px); }
  button{ width:195px;}
 </style></head>
<body><div align="center">
 <button id="myBtnNormal">正常显示图像</button>
 <button id="myBtnBlur">使用高斯模糊滤镜显示图像</button></div>
<svg width="100%" height="100%"><defs><filter id="f1" x="0" y="0">
   <feGaussianBlur in="SourceGraphic" stdDeviation="3"/></filter></defs>
  <image xlink:href="img/A186.jpg" id="MyImage" x="0" y="0" width="400" height="250" filter=
"url(#f1)"/></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< filter >标签用来定义 SVG 滤镜，必需的 id 属性定义向图像应用哪个滤镜。< filter >标签必须嵌套在< defs >标签内，< defs >标签是

definitions 的缩写，它允许对滤镜等特殊元素进行定义。filter:url 属性用来把元素链接到滤镜，当链接滤镜 id 时必须使用＃字符。滤镜效果是通过<feGaussianBlur>标签进行定义的，fe 后缀可用于所有的滤镜。<feGaussianBlur>标签的 stdDeviation 属性可定义模糊效果的程度。

此实例的源文件名是 myHtmlA186.html。

182 在 SVG 中使用矩阵滤镜旋转图像的色相

此实例主要通过在 SVG 中使用 feColorMatrix 滤镜改变图像的饱和度和调整图像的色相。当在 Google Chrome 浏览器中显示该页面时，单击“使用 feColorMatrix 滤镜进行色相旋转”按钮，则图像的色相经过重置之后的效果如图 182-1 所示；单击“使用 feColorMatrix 滤镜调整饱和度”按钮，则图像的饱和度在经过重置之后的效果如图 182-2 所示。有关此实例的主要代码如下。



图 182-1



图 182-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtnHueRotate").click(function() {     //使用 feColorMatrix 滤镜进行色相旋转
     $("svg defs filter feColorMatrix").attr("type", "hueRotate");
     $("svg defs filter feColorMatrix").attr("values", "180");
    });
    $("#myBtnSaturate").click(function() {      //使用 feColorMatrix 滤镜调整饱和度
     $("svg defs filter feColorMatrix").attr("type", "saturate");
     $("svg defs filter feColorMatrix").attr("values", "0.12"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
  button{ width:195px;}
 </style></head>
<body><div align="center">
 <button id="myBtnHueRotate">使用 feColorMatrix 滤镜进行色相旋转</button>
 <button id="myBtnSaturate">使用 feColorMatrix 滤镜调整饱和度</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
```

```
    <!--<feColorMatrix in="SourceGraphic" type="hueRotate" values="180" />-->
    <!--<feColorMatrix in="SourceGraphic" type="saturate" values="0.12" />-->
    <!--<feColorMatrix in="SourceGraphic" type="luminanceToAlpha" />-->
    <feColorMatrix in="SourceGraphic"/></filter></defs>
  <image xlink:href="img/A190.jpg" id="MyImage" x="0" y="0" width="400" height="250" filter="
url(#myFilter)"/></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<feColorMatrix in="SourceGraphic" type="hueRotate" values="180" />表示使用 feColorMatrix 滤镜将图像的色相旋转 180°。在 SVG 中 feColorMatrix 滤镜基于转换矩阵对颜色进行变换，每一像素的颜色值（一个表示为[红，绿，蓝，透明度]的矢量）都经过矩阵乘法（matrix multiplated）计算出的新颜色。除非自定义颜色矩阵，直接设置 in、type、values 几个属性即可实现大部分的颜色矩阵调整功能。

此实例的源文件名是 myHtmlA190.html。

183　在 SVG 中使用混合滤镜叠加两幅图像

此实例主要通过在 SVG 中使用 feBlend 滤镜实现以高亮或深暗的效果混合显示两幅图像。当在 Google Chrome 浏览器中显示该页面时，单击"以深暗模式混合两幅图像"按钮，将在下面以深暗模式混合显示两幅图像，如图 183-1 所示；单击"以高亮模式混合两幅图像"按钮，将在下面以高亮模式混合显示两幅图像，如图 183-2 所示。有关此实例的主要代码如下。



图　183-1



图　183-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnDarken").click(function() {          //以深暗模式混合两幅图像
    $("svg defs filter feBlend").attr("mode", "darken"); });
   $("#myBtnLighten").click(function() {          //以高亮模式混合两幅图像
    $("svg defs filter feBlend").attr("mode", "lighten"); });});
 </script>
 <style type="text/css">
```

```
svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
button{ width:195px;}
</style></head>
<body><div align="center">
<button id="myBtnDarken">以深暗模式混合两幅图像</button>
<button id="myBtnLighten">以高亮模式混合两幅图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
  <!--<feImage xlink:href="img/A126.jpg" width="400" height="250" result="A126" />-->
  <feImage xlink:href="img/B003.jpg" width="400" height="250" result="B003" />
  <!--<feBlend mode="normal" in="A126" in2="B003" />-->
  <feBlend mode="normal" in="SourceGraphic" in2="B003" /></filter></defs>
  <image xlink:href="img/A126.jpg" id="MyImage" x="0" y="0" width="400" height="250" filter=
"url(#myFilter)"/></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<feBlend mode="normal" in="SourceGraphic" in2="B003" />用于设置 feBlend 滤镜的混合模式为 normal。feBlend 滤镜支持的混合模式有 normal、multiply、screen、darken、lighten 等。

此实例的源文件名是 myHtmlA189.html。

184　使用 SVG 滤镜线性校正图像中的像素

此实例主要通过在 SVG 中使用 feComponentTransfer 滤镜对图像的颜色、亮度、对比度等进行调整。当在 Google Chrome 浏览器中显示该页面时，单击“使用 feComponentTransfer 滤镜处理图像一”按钮，则图像经过特效处理之后的效果如图 184-1 所示；单击“使用 feComponentTransfer 滤镜处理图像二”按钮，则图像经过特效处理之后的效果如图 184-2 所示。有关此实例的主要代码如下。



图　184-1



图　184-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnGamma").click(function() {          //使用 feComponentTransfer 滤镜处理图像一
```

```
        $("image").attr("filter", "url(#myFilterGamma)"); });
       $("#myBtnLinear").click(function() {          //使用 feComponentTransfer 滤镜处理图像二
        $("image").attr("filter", "url(#myFilterLinear)"); });});
    </script>
    <style type="text/css">
     svg { margin-top: 10px; position: absolute; left: calc(50% - 200px); }
     button{ width:195px;}
    </style></head>
   <body><div align="center">
    <button id="myBtnGamma">使用 feComponentTransfer 滤镜处理图像一</button>
    <button id="myBtnLinear">使用 feComponentTransfer 滤镜处理图像二</button></div>
   <svg width="100%" height="100%">
    <defs>
     <filter id="myFilterGamma" filterUnits="objectBoundingBox"
                                   x="0%" y="0%" width="100%" height="100%">
      <feComponentTransfer>
       <feFuncR type="gamma" amplitude="2" exponent="5" offset="0"/>
       <feFuncG type="gamma" amplitude="2" exponent="3" offset="0"/>
       <feFuncB type="gamma" amplitude="2" exponent="1" offset="0"/>
      </feComponentTransfer></filter>
     <filter id="myFilterLinear" filterUnits="objectBoundingBox"
                                   x="0%" y="0%" width="100%" height="100%">
      <feComponentTransfer>
       <feFuncR type="linear" slope=".5" intercept=".25"/>
       <feFuncG type="linear" slope=".5" intercept="0"/>
       <feFuncB type="linear" slope=".5" intercept=".5"/>
      </feComponentTransfer></filter></defs>
     <image xlink:href="img/A191.jpg" id="MyImage" x="0" y="0" width="400" height="250" /></svg>
   </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feComponentTransfer 滤镜用于执行亮度、对比度、颜色平衡和阈值等数据的 component-wise 重映射。在定义这些参数时通常需要设置 feFuncR、feFuncG、feFuncB、feFuncA 的值。

此实例的源文件名是 myHtmlA191.html。

185 使用 SVG 滤镜实现马赛克风格的图像

此实例主要在 SVG 的 feMorphology 滤镜中设置不同的 radius 属性值，从而使图像产生不同粗细的马赛克模糊特效。当在 Google Chrome 浏览器中显示该页面时将显示一幅未经滤镜处理的图像，如图 185-1 所示；单击“使用 feMorphology 滤镜实现细马赛克特效”按钮，图像产生的细马赛克效果如图 185-2 所示；单击“使用 feMorphology 滤镜实现粗马赛克特效”按钮，图像产生的粗马赛克效果如图 185-3 所示。有关此实例的主要代码如下。

```
   <!doctype html><html><head><meta charset="UTF-8">
    <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
      $(function() {
       $("#myBtnThin").click(function() {          //使用 feMorphology 滤镜实现细马赛克特效
        $("svg defs filter feMorphology").attr("radius", "1");
       });
```

```
    $("#myBtnFatten").click(function() {         //使用 feMorphology 滤镜实现粗马赛克特效
     $("svg defs filter feMorphology").attr("radius", "5"); });});
  </script>
  <style type="text/css">
   svg { margin-top: 10px;position: absolute;left: calc(50% - 200px);}
   button { width: 195px;}
  </style></head>
 <body>
 <div align="center">
  <button id="myBtnThin">使用 feMorphology 滤镜实现细马赛克特效</button>
  <button id="myBtnFatten">使用 feMorphology 滤镜实现粗马赛克特效</button></div>
 <svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
  <feMorphology operator="dilate" in="SourceGraphic" radius="0" />
  </filter></defs>
   <image xlink:href="img/A192.jpg" id="MyImage" x="0" y="0" width="400" height="250" filter=
 "url(#myFilter)"/></svg>
 </body></html>
```



图 185-1



图 185-2



图 185-3

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feMorphology operator＝"dilate" in＝"SourceGraphic" radius＝"0" />用于设置 feMorphology 滤镜，当以不同的值设置 radius 属性值时会产生不同的粗细效果，产生的马赛克实际上是两倍于 radius 属性值（例如 $("svg defs filter feMorphology").attr("radius", "5")）的矩形。在 SVG 中，feMorphology 滤镜通过操控 alpha 通道实现图像变粗或变细的特效。

此实例的源文件名是 myHtmlA192.html。

186 使用 SVG 滤镜为不规则图像添加阴影

此实例主要通过使用 SVG 的 feGaussianBlur、feOffset、feFlood、feComposite 等滤镜为不规则的图形图像添加阴影特效。当在 Google Chrome 浏览器中显示该页面时将显示一幅没有阴影的不规则 png 图像，单击“显示添加阴影的不规则图像”按钮，则不规则图像添加阴影的效果如图 186-1 所示。有关此实例的主要代码如下。

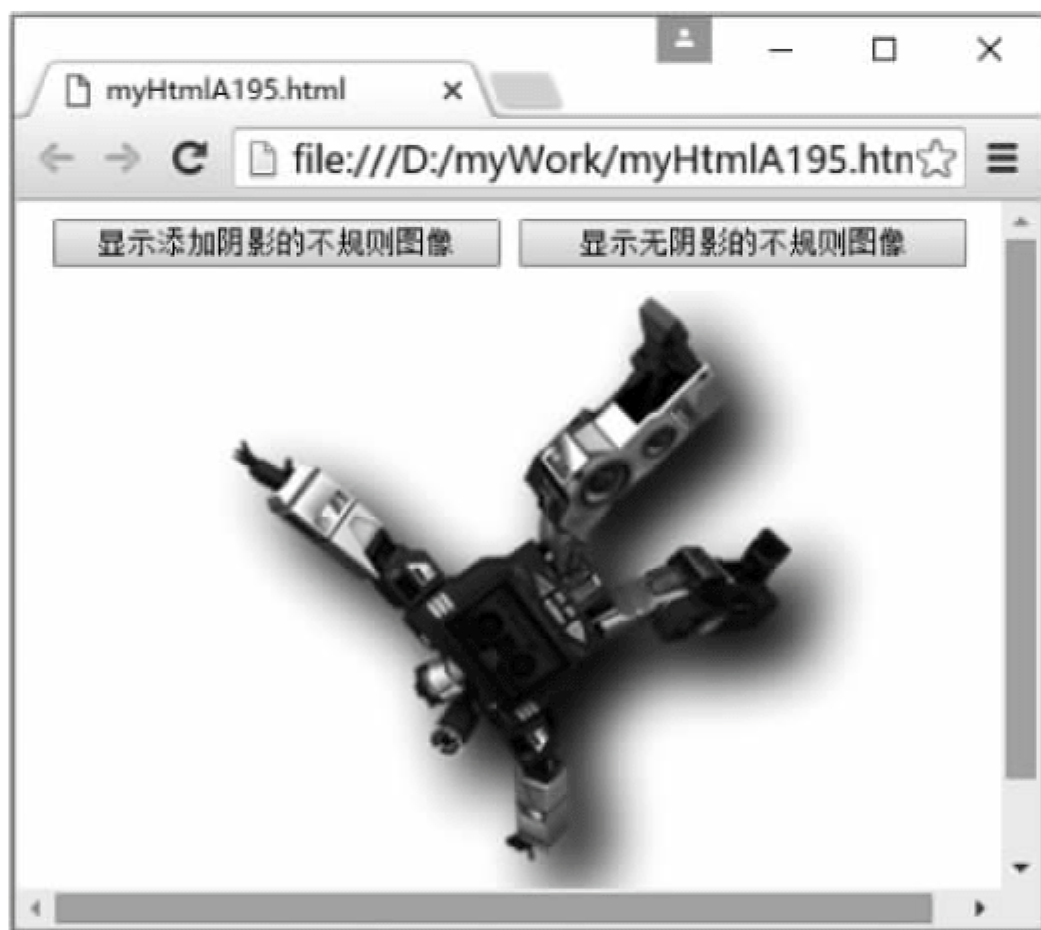


图　186-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script type="text/javascript">
   $(function() {
    $("#myBtnShadow").click(function() {//显示添加阴影的不规则图像
     $("image").attr("filter", "url(#myFilter)");
   });
    $("#myBtnNone").click(function() {//显示无阴影的不规则图像
     $("image").attr("filter", "");
  });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
  button {width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtnShadow">显示添加阴影的不规则图像</button>
```

```
<button id = "myBtnNone">显示无阴影的不规则图像</button></div>
<svg width = "100 % " height = "100 % "><defs>
  <filter id = "myFilter" filterUnits = "objectBoundingBox" x = " - 10 % " y = " - 10 % " width = "150 % "
height = "150 % ">
   <! -- 产生模糊扩散特效 -->
   <feGaussianBlur in = "SourceAlpha" stdDeviation = "13" result = "blurredAlpha"/>
   <! -- 产生阴影偏移特效 -->
   <feOffset in = "blurredAlpha" dx = "14" dy = "14" result = "offsetBlurredAlpha"/>
   <feFlood result = "flooded" style = "flood - color:black;flood - opacity:0.95"/>
   <! -- < feComposite in = " flooded" operator = " atop" in2 = " offsetBlurredAlpha" result =
"coloredShadow"/> -->
   <! -- 设置 in 和 in2 的组合模式 -->
   <feComposite in = "flooded" operator = "in" in2 = "offsetBlurredAlpha" result = "coloredShadow"/>
   <feComposite in = "SourceGraphic" in2 = "coloredShadow" operator = "over"/>
  </filter></defs>
  <image xlink:href = "img/A195.png" id = "MyImage" x = "0" y = "0" width = "400" height = "250" /></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feGaussianBlur、feOffset、feFlood、feComposite 等滤镜组合在一起实现了与 CSS3 的 drop-shadow 滤镜类似的功能，由于这些滤镜的参数众多，所以比 drop-shadow 滤镜具有更高的灵活性和可扩展性。< feGaussianBlur in = "SourceAlpha" stdDeviation = "13" result = "blurredAlpha"/>中的 stdDeviation = "13"表示模糊半径，此值越大模糊范围越大；< feOffset in = "blurredAlpha" dx = "14" dy = "14" result = "offsetBlurredAlpha"/>中的 dx、dy 表示阴影偏移值，可以为负数，正数产生的阴影在右下角，负数产生的阴影在左上角。

此实例的源文件名是 myHtmlA195.html。

187 使用 SVG 滤镜为图像添加翘边的阴影

此实例主要通过使用 SVG 的 feGaussianBlur 滤镜为图像添加翘边的阴影特效。当在 Google Chrome 浏览器中显示该页面时，单击"显示原始的图像"按钮，将在下面显示原始的图像；单击"显示翘边阴影的图像"按钮，图像在添加翘边的阴影特效之后的效果如图 187-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script type = "text/javascript">
  $ (function() {
   $ ("#myBtn1").click(function() {        //显示原始的图像
    $ ("image").show();
    $ ("path").hide();
   });
   $ ("#myBtn2").click(function() {        //显示翘边阴影的图像
    $ ("image").show();
    $ ("path").show();
  });});
 </script>
 <style type = "text/css">
  svg { margin - top: 10px; position: absolute;
        left: calc(50 % - 200px);}
```

```
    button { width: 195px;}
  </style></head>
<body><div align="center">
  <button id="myBtn1">显示原始的图像</button>
  <button id="myBtn2">显示翘边阴影的图像</button>
</div>
<svg width="100%" height="100%"><defs>
  <filter width="440" height="290" x="0" y="0" id="myFilter" filterUnits="userSpaceOnUse">
   <feGaussianBlur stdDeviation="5"/></filter></defs>
  <path d="M30 30 L430 30 Q410 155 430 280 L30 280 Q50 155 30 30" fill="#000" filter=url
("#myFilter") transform="translate(-20,-20)"/>
  <image x="0" y="0" width="400" height="250" xlink:href="img/A211.jpg" transform=
"scale(0.96),translate(19,14)"/></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feGaussianBlur 滤镜用于为元素添加模糊特效，需要注意它只有一个参数——stdDeviation，该参数控制模糊程度，数字越大越模糊，数字为 0 则为其原始状态。如果写成<feGaussianBlur stdDeviation="1 15"/>，则 1 代表水平方向的模糊半径，15 代表垂直方向的模糊半径。<path d="M30 30 L430 30 Q410 155 430 280 L30 280 Q50 155 30 30" fill="#000" filter=url("#myFilter") transform="translate(-20,-20)"/>用于创建一个带阴影效果的多边形，如图 187-2 所示，图像则是直接叠加在此阴影之上，因为多边形的左、右两端带有弧线，所以在叠加图像之后有翘边的阴影效果。

此实例的源文件名是 myHtmlA213.html。



图 187-1

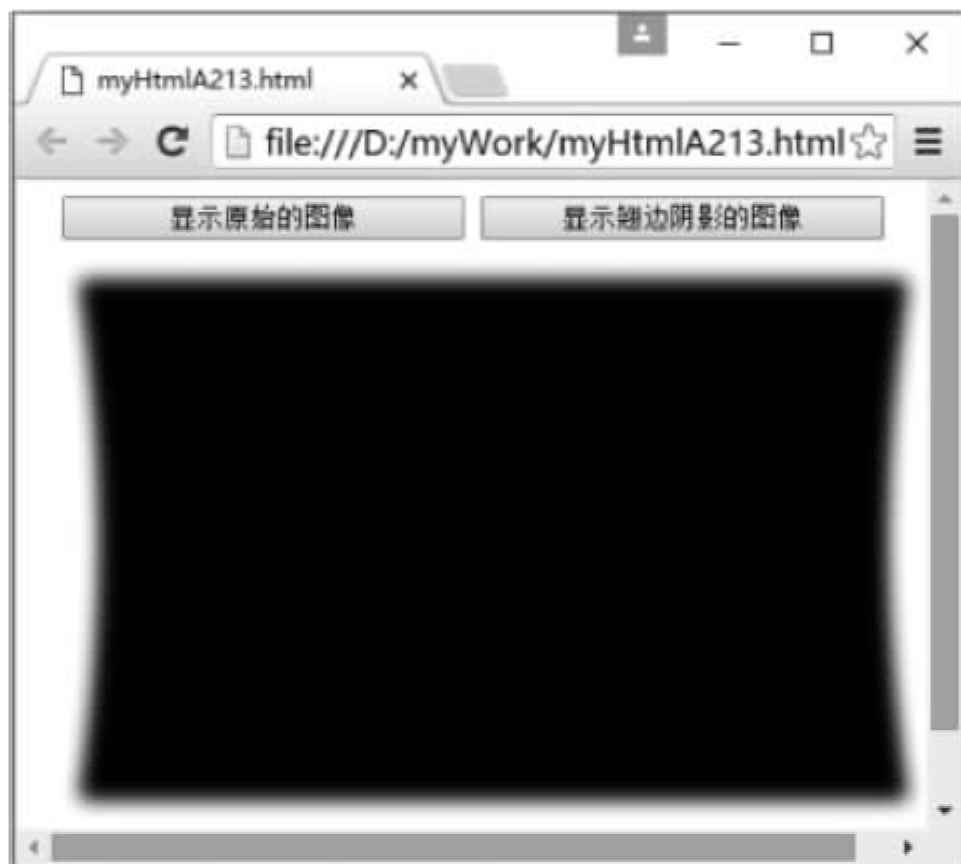


图 187-2

188 使用 SVG 的 feOffset 滤镜生成悬空阴影

此实例主要通过在 SVG 中使用 feOffset 滤镜为图像生成悬空阴影。当在 Google Chrome 浏览器中显示该页面时，单击“使用青色光源照射图像”按钮，为图像生成的悬空阴影如图 188-1 所示；单击“使用黄色光源照射图像”按钮，为图像生成的悬空阴影如图 188-2 所示。有关此实例的主要代码如下。



图 188-1

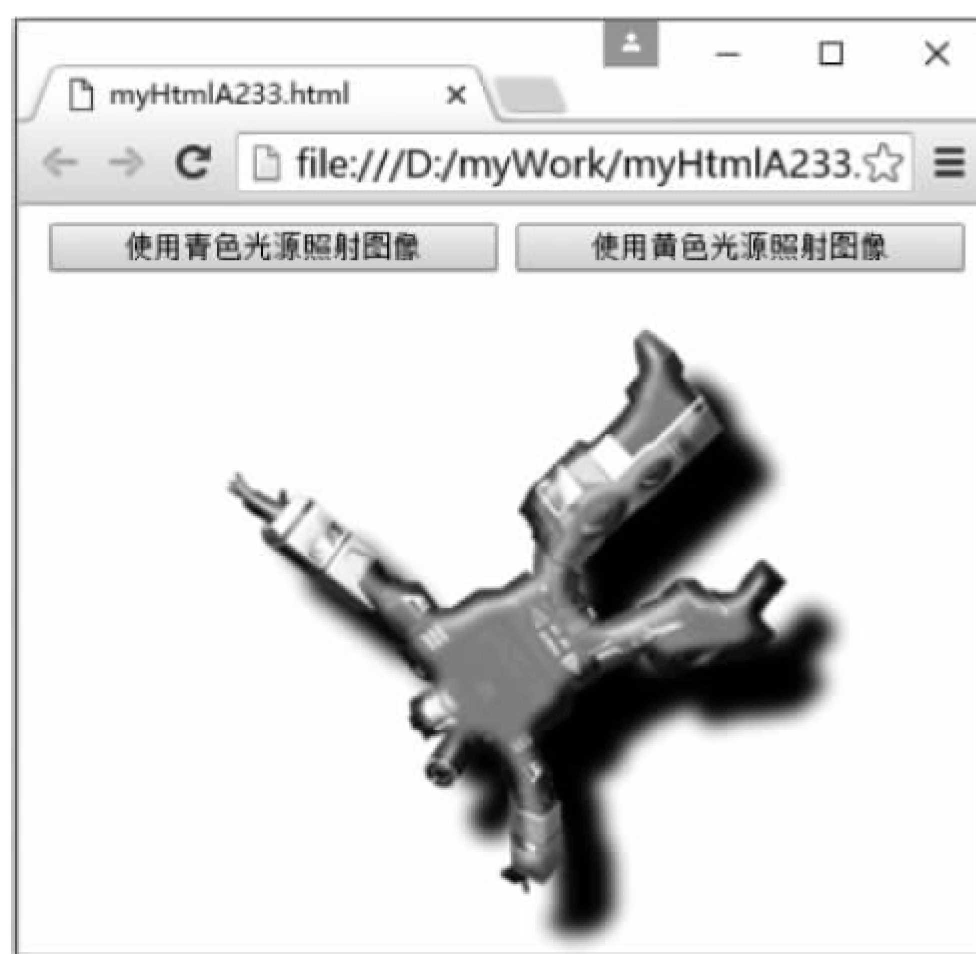


图 188-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#myBtn1").click(function(){//使用青色光源照射图像
    $("svg defs filter feSpecularLighting").attr("lighting-color", "cyan");
    $("g").attr("filter", "url(#myFilter)");
   });
   $("#myBtn2").click(function(){//使用黄色光源照射图像
    $("svg defs filter feSpecularLighting").attr("lighting-color", "yellow");
    $("g").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 20px;position: absolute;left: calc(50% - 200px);}
  div { margin-top: 2px; margin-left: 2px;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">使用青色光源照射图像</button>
 <button id="myBtn2">使用黄色光源照射图像</button></div>
<div><svg width="400" height="275">
    <defs><filter id="myFilter">
    <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
    <feOffset in="blur" dx="21" dy="21" result="offsetBlur"/>
    <feSpecularLighting in="blur" surfaceScale="5" specularConstant=".75" specularExponent="20"
lighting-color="#bbbbbb" result="specOut">
     <fePointLight x="-5000" y="-10000" z="20000"/></feSpecularLighting>
    <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut"/>
    <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic" k1="0" k2="1" k3="1"
k4="0" result="litPaint"/>
   <feMerge><feMergeNode in="offsetBlur"/><feMergeNode in="litPaint"/></feMerge>
   </filter></defs>
   <g><image id="myImage" x="0" y="0" width="400px" height="250px" xlink:href="img/A195.png">
</image></g></svg></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，生成悬空阴影主要由 feOffset 滤镜实现，可以通过设置 feOffset 滤镜的 dx 属性值和 dy 属性值调整悬空阴影的偏移量大小，如果其值为负数，则产生的悬空方向相反；feGaussianBlur 滤镜主要用于为阴影产生模糊效果。

此实例的源文件名是 myHtmlA233.html。

189 使用 SVG 滤镜对图像进行暗化和亮化

此实例主要通过在 SVG 的 feComposite 滤镜中使用不同的数学算法使图像产生暗化特效和亮化特效。当在 Google Chrome 浏览器中显示该页面时，单击“使用 feComposite 滤镜暗化图像”按钮，图像产生的暗化效果如图 189-1 所示；单击“使用 feComposite 滤镜亮化图像”按钮，图像产生的亮化效果如图 189-2 所示。有关此实例的主要代码如下。



图 189-1



图 189-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("#myBtnDark").click(function(){//使用 feComposite 滤镜暗化图像
    $("image").attr("filter", "url(#myFilter1)");
   });
   $("#myBtnLight").click(function(){//使用 feComposite 滤镜亮化图像
    $("image").attr("filter", "url(#myFilter2)"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtnDark">使用 feComposite 滤镜暗化图像</button>
 <button id="myBtnLight">使用 feComposite 滤镜亮化图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter1">
   <feComposite operator="arithmetic" k1="1" k2="0" k3="0" k4="0"/></filter>
  <filter id="myFilter2"><feComposite operator="arithmetic" k1="0" k2="1" k3="1"
k4="0"/></filter></defs>
```

```
<image xlink:href="img/A193.jpg" id="MyImage" x="0" y="0" width="400" height="250" /></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<feComposite operator="arithmetic" k1="1" k2="0" k3="0" k4="0"/>用于设置 feComposite 滤镜以产生暗化特效，<feComposite operator="arithmetic" k1="0" k2="1" k3="1" k4="0"/>用于设置 feComposite 滤镜以产生亮化特效。当 operator 属性值为 arithmetic 时，通常应该设置 k1、k2、k3、k4 这 4 个与数学算法密切相关的属性。例如，当<feComposite operator="arithmetic" k1="0" k2="5" k3="5" k4="0"/>时亮化效果将特别明显。

此实例的源文件名是 myHtmlA193.html。

190 使用 SVG 滤镜对图像进行轻微模糊处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行轻微模糊处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"显示轻微的模糊图像"按钮，图像被 feConvolveMatrix 滤镜处理后的效果如图 190-1 所示。有关此实例的主要代码如下。



图 190-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的原始图像
     $("image").attr("filter", "");
   });
   $("#myBtn2").click(function() {//显示轻微的模糊图像
     $("image").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
```

```
<button id = "myBtn1">显示默认的原始图像</button>
 <button id = "myBtn2">显示轻微的模糊图像</button></div>
<svg width = "100 % " height = "100 % "><defs><filter id = "myFilter" x = "0" y = "0">
   <feConvolveMatrix kernelMatrix = "1 1 1 1 1 1 1 1 1" >
   </feConvolveMatrix></filter></defs>
  <image xlink:href = "img/A199.jpg" id = "MyImage" x = "0" y = "0" width = "400" height =
"250"/></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix="1 1 1 1 1 1 1 1 1" >中的 kernelMatrix 属性值是一个 3 × 3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生模糊特效。

此实例的源文件名是 myHtmlA199.html。

191 使用 SVG 滤镜对图像进行深度模糊处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行深度模糊处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示深度的模糊图像”按钮，图像被 feConvolveMatrix 滤镜处理后的效果如图 191-1 所示。有关此实例的主要代码如下。



图 191-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
   $ (function() {
    $ ("#myBtn1").click(function() {//显示默认的原始图像
     $ ("image").attr("filter", "");
    });
    $ ("#myBtn2").click(function() {//显示深度的模糊图像
     $ ("image").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type = "text/css">
  svg { margin - top: 10px;position: absolute;left: calc(50 % - 200px);}
  button { width: 195px;}
 </style></head>
```

```
<body><div align="center">
 <button id="myBtn1">显示默认的原始图像</button>
 <button id="myBtn2">显示深度的模糊图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
   <feConvolveMatrix kernelMatrix="3 0 3 0 0 0 3 0 3" >
   </feConvolveMatrix></filter></defs>
   <image xlink:href="img/A200.jpg" id="MyImage" x="0" y="0" width="400" height="250"/></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix="3 0 3 0 0 0 3 0 3" >中的 kernelMatrix 属性值是一个 3×3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生深度模糊特效。在一般情况下，对图像进行模糊处理或锐化处理都是通过阵列对相邻像素进行处理来实现的。

此实例的源文件名是 myHtmlA200.html。

192　使用 SVG 滤镜对图像进行加粗锐化处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行加粗锐化处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"显示加粗的锐化图像"按钮，图像被 feConvolveMatrix 滤镜处理后的效果如图 192-1 所示。有关此实例的主要代码如下。



图　192-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtn1").click(function() {//显示默认的原始图像
     $("image").attr("filter", "");
   });
    $("#myBtn2").click(function() {//显示加粗的锐化图像
     $("image").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type="text/css">
```

```
   svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
   button { width: 195px;}
  </style></head>
 <body><div align="center">
  <button id="myBtn1">显示默认的原始图像</button>
  <button id="myBtn2">显示加粗的锐化图像</button></div>
 <svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
    <feConvolveMatrix kernelMatrix="0 -1 0 -1 5 -1 0 -1 0">
    </feConvolveMatrix></filter></defs>
   <image xlink:href="img/A201.jpg" id="MyImage" x="0" y="0" width="400" height="250"/></svg>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<feConvolveMatrix kernelMatrix="0 −1 0 −1 5 −1 0 −1 0">中的 kernelMatrix 属性值是一个 3 × 3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生加粗锐化特效。在 SVG 中，feConvolveMatrix 不仅能够锐化图像，同时也能模糊化图像，这主要取决于 kernelMatrix 属性值所代表的阵列；feGaussianBlur 滤镜通常只能模糊图像。

此实例的源文件名是 myHtmlA201.html。

193　使用 SVG 滤镜对图像进行加亮锐化处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行加亮锐化处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示加亮的锐化图像”按钮，图像被 feConvolveMatrix 滤镜加亮锐化处理后的效果如图 193-1 所示。有关此实例的主要代码如下。



图　193-1

```
 <!doctype html><html><head><meta charset="UTF-8">
  <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtn1").click(function() {//显示默认的原始图像
     $("image").attr("filter", "");
    });
```

```
    $("#myBtn2").click(function() {//显示加亮的锐化图像
     $("image").attr("filter", "url(#myFilter)"); });});
  </script>
  <style type="text/css">
   svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
   button { width: 195px;}
  </style></head>
 <body><div align="center">
  <button id="myBtn1">显示默认的原始图像</button>
  <button id="myBtn2">显示加亮的锐化图像</button></div>
 <svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
   <feConvolveMatrix kernelMatrix="0 -2 0 -2 15 -2 0 -2 0">
   </feConvolveMatrix></filter></defs>
  <image xlink:href="img/A202.jpg" id="MyImage" x="0" y="0" width="400" height= "250"/>
 </svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix="0 −2 0 −2 15 −2 0 −2 0">中的 kernelMatrix 属性值是一个 3×3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生加亮锐化特效。kernelMatrix 属性值所代表的 3×3 像素阵列主要是中心那个值和周边的 8 个值，用不同的数字测试会使图像获得不同的特效。

此实例的源文件名是 myHtmlA202.html。

194 使用 SVG 滤镜对图像进行边缘化处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行边缘化处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示边缘化的图像”按钮，图像被 feConvolveMatrix 滤镜边缘化处理后的效果如图 194-1 所示。有关此实例的主要代码如下。

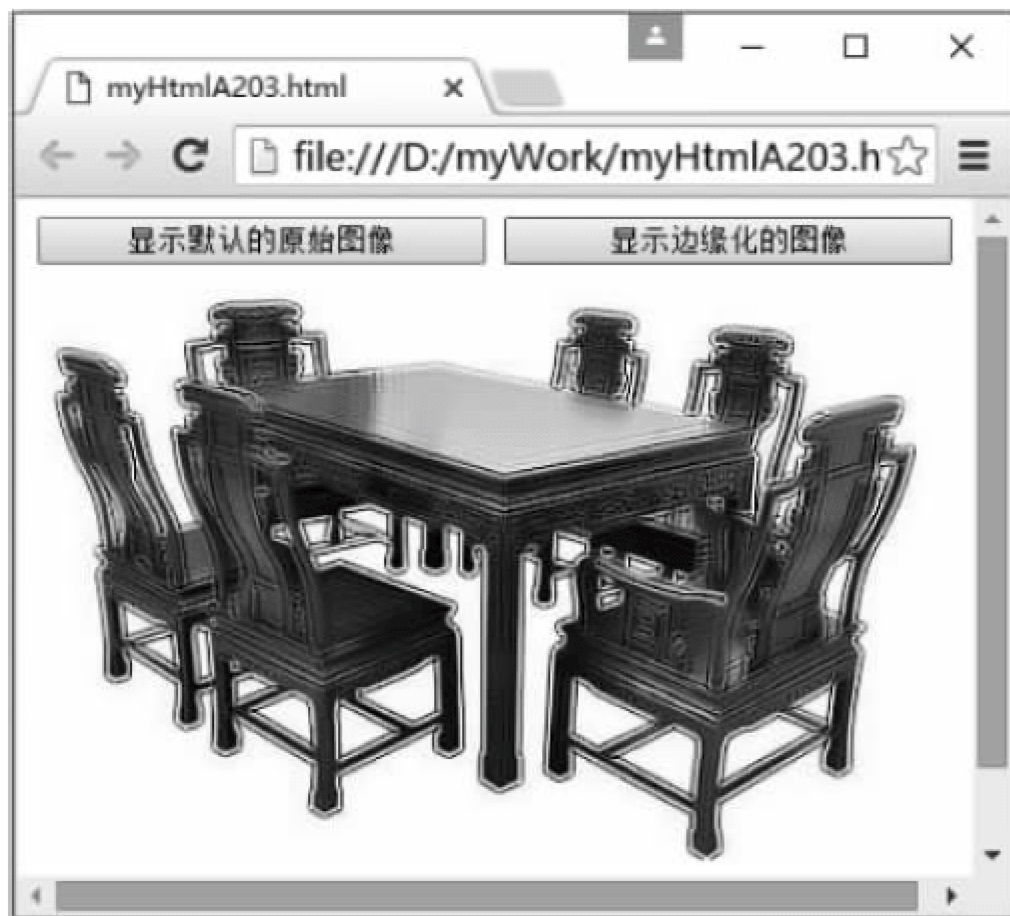


图 194-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
```

```
    $(function() {
     $("#myBtn1").click(function() {//显示默认的原始图像
      $("image").attr("filter", "");
     });
     $("#myBtn2").click(function() {//显示边缘化的图像
      $("image").attr("filter", "url(#myFilter)"); });});
  </script>
  <style type="text/css">
   svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
   button { width: 195px; }
  </style></head>
 <body><div align="center">
  <button id="myBtn1">显示默认的原始图像</button>
  <button id="myBtn2">显示边缘化的图像</button></div>
 <svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
    <feConvolveMatrix kernelMatrix=" 1 1 1 1 -6 1 1 1 1">
    </feConvolveMatrix></filter></defs>
   <image xlink:href="img/A202.jpg" id="MyImage" x="0" y="0" width="400" height="250"/></svg>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix=" 1 1 1 1 -6 1 1 1 1" >中的 kernelMatrix 属性值是一个 3×3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生边缘化特效。

此实例的源文件名是 myHtmlA203.html。

195 使用 SVG 滤镜对图像进行浮雕特效处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行浮雕特效处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"显示浮雕效果的图像"按钮，图像被 feConvolveMatrix 滤镜浮雕处理后的效果如图 195-1 所示。有关此实例的主要代码如下。



图 195-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的原始图像
    $("image").attr("filter", "");
   });
   $("#myBtn2").click(function() {//显示浮雕效果的图像
    $("image").attr("filter", "url(#myFilter)");
  });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示默认的原始图像</button>
 <button id="myBtn2">显示浮雕效果的图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
   <feConvolveMatrix kernelMatrix=" -2 -1 0 -1 1 1 0 1 2">
   </feConvolveMatrix></filter></defs>
 <image xlink:href="img/A204.jpg" id="MyImage" x="0" y="0" width="400" height="250"/>
</svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix=" −2 −1 0 −1 1 1 0 1 2">中的 kernelMatrix 属性值是一个 3×3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生浮雕特效。

此实例的源文件名是 myHtmlA204.html。

196　使用 SVG 滤镜对图像进行木刻特效处理

此实例主要通过使用 SVG 的 feConvolveMatrix 滤镜对图像进行木刻特效处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示木刻效果的图像”按钮，图像被 feConvolveMatrix 滤镜木刻处理后的效果如图 196-1 所示。有关此实例的主要代码如下。



图　196-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script type="text/javascript">
   $(function() {
    $("#myBtn1").click(function() { //显示默认的原始图像
     $("image").attr("filter", "");
    });
    $("#myBtn2").click(function() { //显示木刻效果的图像
     $("image").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示默认的原始图像</button>
 <button id="myBtn2">显示木刻效果的图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter" x="0" y="0">
    <feConvolveMatrix kernelMatrix="-9 -2 0 -2 1 2 0 2 9">
    </feConvolveMatrix></filter></defs>
  <image xlink:href="img/A204.jpg" id="MyImage" x="0" y="0" width="400" height="250"/></svg></
body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< feConvolveMatrix kernelMatrix="-9 -2 0 -2 1 2 0 2 9">中的 kernelMatrix 属性值是一个 3×3 的像素阵列，feConvolveMatrix 滤镜使用此像素阵列对原始图像的像素值重新进行计算以产生木刻特效。

此实例的源文件名是 myHtmlA205.html。

197 使用 SVG 滤镜为图像添加波纹扩散特效

此实例主要通过使用 SVG 的 feDisplacementMap 滤镜对图像添加波纹扩散的特效。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"显示波纹扩散的图像"按钮，图像添加波纹扩散效果之后如图 197-1 所示。有关此实例的主要代码如下。

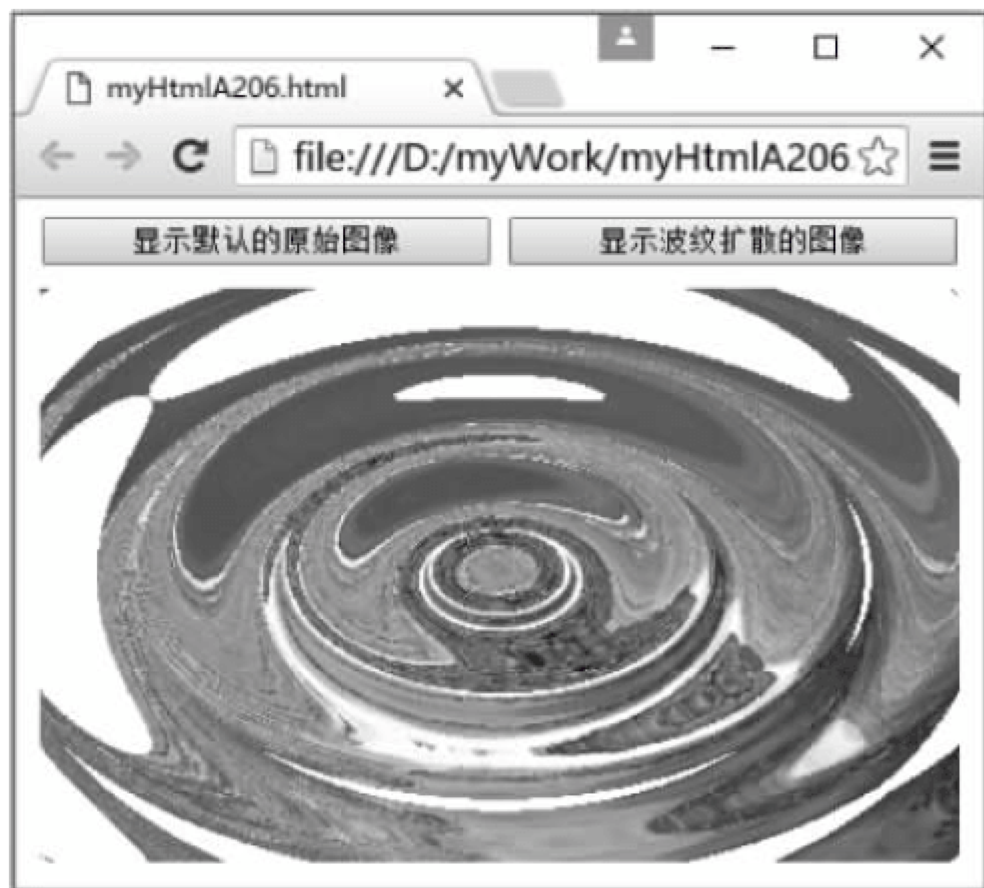


图 197-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的原始图像
    $("#myImage").show(); $("rect").hide();
   });
   $("#myBtn2").click(function() {//显示波纹扩散的图像
    $("#myImage").hide(); $("rect").show(); });});
 </script><style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center"><button id="myBtn1">显示默认的原始图像</button>
 <button id="myBtn2">显示波纹扩散的图像</button></div>
<svg width="400" height="250"><defs>
  <filter id="myFilter" primitiveUnits="objectBoundingBox" x="0" y="0" width="1.1" height="1.1">
   <feImage result="pict1" xlink:href="#m1" x="0" y="0" width="1.1" height="1.1">
</feImage>
   <feImage result="pict2" xlink:href="#m2" x="0" y="0" width="1.1" height="1.1">
</feImage>
   <feDisplacementMap id="fdm" scale="0.25" xChannelSelector="R" yChannelSelector="R" in2=
"pict2" in="pict1"></feDisplacementMap></filter>
  <!--cx cy是圆心坐标-->
  <radialGradient id="g" cx="0.5" cy="0.5" r="0.08" spreadMethod="reflect">
   <stop offset="0" stop-color="black"></stop>
   <stop offset="1" stop-color="white"></stop></radialGradient>
  <rect id="m2" x="0" y="0" width="400px" height="250px" fill="url(#g)"></rect>
  <image id="m1" x="0" y="0" width="400px" height="250px" xlink:href="img/A206.jpg"></image>
</defs>
 <rect x="0" y="0" width="400px" height="250px" filter="url(#myFilter)"></rect>
  <image id="myImage" x="0" y="0" width="400px" height="250px" xlink:href="img/A206.jpg">
</image></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<feDisplacementMap id="fdm" scale="0.25" xChannelSelector="R" yChannelSelector="R" in2="pict2" in="pict1">主要是根据指定的属性值使用 in2="pict2"的(放射渐变模型)像素置换 in="pict1"的(原始)像素，从而使 in="pict1"产生波纹扩散的效果。

此实例的源文件名是 myHtmlA206.html。

198 使用 SVG 滤镜为图像添加波纹起伏特效

此实例主要通过使用 SVG 的 feDisplacementMap 滤镜对图像添加波纹起伏的特效。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"显示波纹起伏的图像"按钮，添加波纹起伏效果的图像如图 198-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的原始图像
    $("#myImage").show(); $("rect").hide();
```

```
      });
      $("#myBtn2").click(function() {//显示波纹起伏的图像
        $("#myImage").hide(); $("rect").show();
    });});
   </script><style type="text/css">
    svg { margin-top: 10px; position: absolute; left: calc(50% - 200px); }
    button { width: 195px; }
   </style></head>
 <body><div align="center">
   <button id="myBtn1">显示默认的原始图像</button>
   <button id="myBtn2">显示波纹起伏的图像</button></div>
 <svg width="400" height="250">
   <defs>
    <filter id="f" primitiveUnits="objectBoundingBox" x="0" y="0" width="1.1" height="1.1">
      <feImage result="pict1" xlink:href="#m1" x="0" y="0" width="1.1" height="1.1">
 </feImage>
      <feImage result="pict2" xlink:href="#m2" x="0" y="0" width="1.1" height="1.1">
 </feImage>
      <feDisplacementMap id="fdm" scale="0.05" xChannelSelector="A" yChannelSelector="A" in2=
 "pict2" in="pict1"></feDisplacementMap></filter>
    <linearGradient id="g" x1="0" y1="0" x2="0.2" y2="0.02" spreadMethod="reflect">
      <stop offset="0" stop-color="rgba(0,0,0,0.8)"></stop>
      <stop offset="0.2" stop-color="rgba(0,0,0,0.7)"></stop>
      <stop offset="0.6" stop-color="rgba(0,0,0,0.1)"></stop>
      <stop offset="1" stop-color="rgba(0,0,0,0)"></stop></linearGradient>
    <rect id="m2" x="0" y="0" width="400" height="250" fill="url(#g)"></rect>
    <image id="m1" x="0" y="0" width="400" height="250" xlink:href="img/A206.jpg">
 </image></defs>
    <rect x="0" y="0" width="400" height="250" filter="url(#f)"></rect>
    <rect x="0" y="0" width="400" height="250" fill="url(#g2)" ></rect>
    <image id="myImage" x="0" y="0" width="400px" height="250px" xlink:href="img/A206.jpg">
 </image></svg>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feDisplacementMap 滤镜主要是通过像素置换实现波纹起伏特效，它主要使用下列公式置换像素。



图 198-1

P'(x,y) <- P(x+scale * (XC(x,y)−0.5), y+scale * (YC(x,y)−0.5))

大概意思是,P'(x,y)是置换之后的像素坐标,公式就是原本的坐标(x,y)加上振幅比例(scale)与 xChannelSelector 或 yChannelSelector 的乘积。

简单来说就是提供一张以 R 或 G、B、A 为主的图片,目的是作为另外一张图片的置换参考,怎样置换取决于提供的图片颜色以及被置换的图片颜色,如果以 R 通道为例,越红置换的比例越高,但如果图片里面没有红色,则 R 作为置换的效果就不明显或与其他颜色通道不同。该实例则主要通过 A 通道解决此问题。注意观察,< stop offset="0" stop-color="rgba(0,0,0,0.8)"></stop >中的 0.8 就是 A 值,修改 4 个< stop >标签的任意 A 值就会得到不同的起伏特效。

此实例的源文件名是 myHtmlA207.html。

199 使用 SVG 的放射渐变创建 3D 风格的球体

此实例主要通过使用 SVG 的放射渐变 radialGradient 创建类似于月全食效果的 3D 球体。当在 Google Chrome 浏览器中显示该页面时将显示一个以绿色为主色调的 3D 球体,如图 199-1 所示;单击"显示红色的 3D 球体"按钮,将显示一个类似于月全食的以红色为主色调的 3D 球体,如图 199-2 所示。有关此实例的主要代码如下。



图 199-1



图 199-2

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
   $ (function() {
    $ ("#myBtn1").click(function() {//显示绿色的 3D 球体
      $ ("circle").attr("fill", "url(#myFilterGreen)"); });
    $ ("#myBtn2").click(function() {//显示红色的 3D 球体
      $ ("circle").attr("fill", "url(#myFilterRed)"); });});
 </script>
 <style type = "text/css">
  svg { margin - top: 10px;position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
```

```
</style></head>
<body><div align="center"><button id="myBtn1">显示绿色的 3D 球体</button>
 <button id="myBtn2">显示红色的 3D 球体</button></div>
<svg width="400" height="400"><defs>
  <radialGradient id="myFilterGreen" cx="0.3" cy="0.3" r="0.7">
   <stop offset="0%" stop-color="#FFF"></stop>
   <stop offset="30%" stop-color="#9F9"></stop>
   <stop offset="70%" stop-color="#373"></stop>
   <stop offset="100%" stop-color="#000"></stop></radialGradient>
  <radialGradient id="myFilterRed" cx="0.3" cy="0.3" r="0.7">
   <stop offset="0%" stop-color="#FFF"></stop>
   <stop offset="30%" stop-color="yellow"></stop>
   <stop offset="70%" stop-color="red"></stop>
   <stop offset="100%" stop-color="#000"></stop></radialGradient></defs>
 <circle cx="200" cy="200" r="180" fill="url(#myFilterGreen)"/></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，放射(径向)渐变 radialGradient 是渐变方向沿着圆形半径的渐变。放射渐变 radialGradient 的 cx 和 cy 属性规定渐变的圆心位置，r 规定渐变的半径，这 3 个属性确定了渐变的区域。放射渐变 radialGradient 常用的<stop>元素规定进行渐变的区域和定义渐变的相关颜色。其中的 offset 属性值可以是百分数，也可以是小数，它和 stop-color 属性一起规定每个渐变点的颜色值，在一般情况下该值都是纯色。

此实例的源文件名是 myHtmlA208.html。

200　使用 SVG 滤镜对图像进行离散特效处理

此实例主要通过使用 SVG 的 feComponentTransfer 滤镜对图像进行离散特效处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示离散特效的图像”按钮，图像经过 feComponentTransfer 滤镜处理后产生的离散效果如图 200-1 所示。有关此实例的主要代码如下。

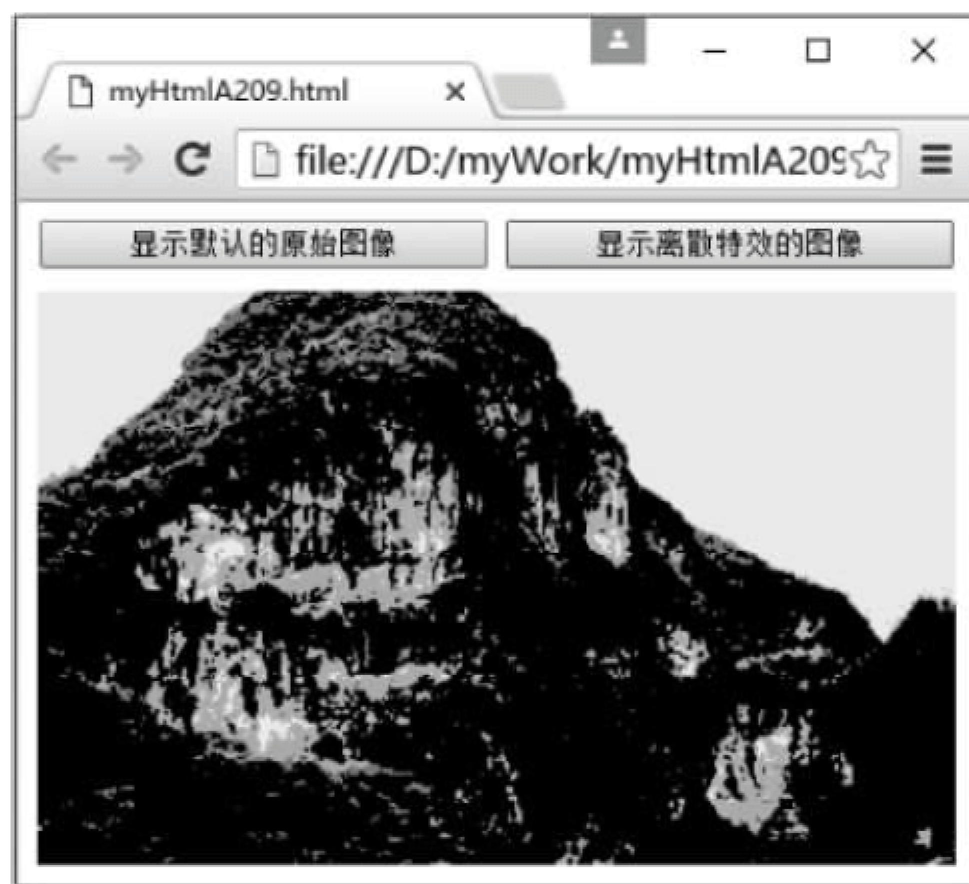


图　200-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的原始图像
    $("image").attr("filter", ""); });
   $("#myBtn2").click(function() { //显示离散特效的图像
    $("image").attr("filter", "url(#myFilter)"); });});
 </script>
 <style type="text/css">
  svg {margin-top: 10px;position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center"><button id="myBtn1">显示默认的原始图像</button>
 <button id="myBtn2">显示离散特效的图像</button></div>
<svg width="400" height="250"><defs>
  <filter id="myFilter" filterUnits="userSpaceOnUse" x="0" y="0" width="400" height="250">
   <feComponentTransfer>
    <feFuncR type="discrete" tableValues="0.0 1.0 1.0 1.0"/>
    <feFuncG type="discrete" tableValues="0.0 0.5 0.5 0.9"/>
    <feFuncB type="discrete" tableValues="0.0 0.6"/>
   </feComponentTransfer></filter></defs>
 <image x="0" y="0" width="400" height="250" xlink:href="img/A209.jpg"/></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feComponentTransfer 滤镜用于针对图像的每个像素通过公式计算调整亮度、对比度等特性，以类似于 Photoshop 的曲线调整模式对图像进行特效处理。feComponentTransfer 滤镜包含 4 个元素，它们分别是 feFuncR、feFuncG、feFuncB、feFuncA，也就是可以针对 R、G、B、A 进行独立调整，调整的类型（type）分别是 identity、table、discrete、linear、gamma。当前实例主要采用 discrete 调整图像。

此实例的源文件名是 myHtmlA209.html。

201　使用 SVG 滤镜以蓝光或红光处理图像

此实例主要通过使用 SVG 的 feComponentTransfer 滤镜对图像进行蓝光或红光特效处理。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“以红光特效显示图像”按钮，则图像经过 feComponentTransfer 滤镜处理后产生的红光效果如图 201-1 所示；单击“以蓝光特效显示图像”按钮，图像经过 feComponentTransfer 滤镜处理后产生的蓝光效果如图 201-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//以红光特效显示图像
    $("image").attr("filter", "url(#myFilterRed)");
   });
   $("#myBtn2").click(function() {//以蓝光特效显示图像
    $("image").attr("filter", "url(#myFilterBlue)"); });});
 </script>
 <style type="text/css">
```

```
    svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
    button { width: 195px;}
   </style></head>
  <body><div align="center">
   <button id="myBtn1">以红光特效显示图像</button>
   <button id="myBtn2">以蓝光特效显示图像</button></div>
  <svg width="400" height="250"><defs>
    <filter id="myFilterRed" filterUnits="userSpaceOnUse" x="0" y="0" width="400" height="250">
     <feComponentTransfer><feFuncR type="table" tableValues="0.99 0.7 0.9 1.0"/>
      <feFuncG type="table" tableValues="0.2 0.7 0.9 1.0"/>
      <feFuncB type="table" tableValues="0.2 0.7 0.9 1.0"/>
      </feComponentTransfer></filter>
    <filter id="myFilterBlue" filterUnits="userSpaceOnUse" x="0" y="0" width="400" height="250">
  <feComponentTransfer>
      <feFuncR type="table" tableValues="0.0 0.7 0.9 1.0"/>
      <feFuncG type="table" tableValues="0.2 0.7 0.9 1.0"/>
      <feFuncB type="table" tableValues="0.99 0.7 0.9 1.0"/>
     </feComponentTransfer></filter></defs>
   <image x="0" y="0" width="400" height="250" xlink:href="img/A210.jpg"/></svg>
  </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feComponentTransfer 滤镜主要通过公式计算对图像像素的亮度、对比度等特性进行调整。feComponentTransfer 滤镜包含 4 个元素，它们分别是 feFuncR、feFuncG、feFuncB、feFuncA，也就是可以针对 R、G、B、A 进行独立调整，调整的类型(type)分别是 identity、table、discrete、linear、gamma。当前实例主要采用 table 调整图像，以实现红光或蓝光特效。

此实例的源文件名是 myHtmlA210.html。



图 201-1



图 201-2

202 使用 SVG 滤镜根据指定颜色消除像素

此实例主要通过使用 SVG 的 feComponentTransfer 滤镜消除图像中指定颜色的像素。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“消除图像的绿色像素”按钮，则图像经过 feComponentTransfer 滤镜处理后产生的效果如图 202-1 所示；单击“消除图像的蓝色像

素”按钮，图像经过 feComponentTransfer 滤镜处理后产生的效果如图 202-2 所示。有关此实例的主要代码如下。



图　202-1



图　202-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//消除图像的绿色像素
    $("image").attr("filter", "url(#myFilterLinearG)");
   });
   $("#myBtn2").click(function() {//消除图像的蓝色像素
    $("image").attr("filter", "url(#myFilterLinearB)"); });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">消除图像的绿色像素</button>
 <button id="myBtn2">消除图像的蓝色像素</button></div>
<svg width="100%" height="100%"><defs>
 <filter id="myFilterLinearB" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%"
height="100%">
  <feComponentTransfer><feFuncB type="linear" slope="0" intercept="0"/>
  </feComponentTransfer></filter>
 <filter id="myFilterLinearG" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%"
height="100%">
  <feComponentTransfer><feFuncG type="linear" slope="0" intercept="0"/>
  </feComponentTransfer></filter></defs>
 <image x="0" y="0" width="400" height="250" xlink:href="img/A211.jpg" /></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feComponentTransfer 滤镜主要通过公式计算对图像的亮度、对比度等特性进行调整。feComponentTransfer 滤镜包含 4 个元素，它们分别是 feFuncR、feFuncG、feFuncB、feFuncA，也就是可以针对 R、G、B、A 进行独立调整，调整的类型

(type)分别是 identity、table、discrete、linear、gamma。当前实例主要采用 linear 调整图像。与 linear 有关的参数有两个，一个是 slope，也就是斜率，预设值为 1；另外一个是 intercept，预设值为 0。linear 的公式为 C'=slope * C+intercept。

此实例的源文件名是 myHtmlA211.html。

203 使用 SVG 滤镜 Gamma 校正图像的像素

此实例主要使用 SVG 的 feComponentTransfer 滤镜，从而实现使用 Gamma 方法校正图像中的像素。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击"Gamma 校正图像的绿色像素"按钮，图像经过 feComponentTransfer 滤镜处理后产生的效果如图 203-1 所示；单击"Gamma 校正图像的红色像素"按钮，图像经过 feComponentTransfer 滤镜处理后产生的效果如图 203-2 所示。有关此实例的主要代码如下。



图 203-1



图 203-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//Gamma 校正图像的绿色像素
    $("image").attr("filter", "url(#myFilterGammaG)");
   });
   $("#myBtn2").click(function() {//Gamma 校正图像的红色像素
    $("image").attr("filter", "url(#myFilterGammaR)");
  });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px;position: absolute;left: calc(50% - 200px);}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">Gamma 校正图像的绿色像素</button>
 <button id="myBtn2">Gamma 校正图像的红色像素</button></div>
<svg width="100%" height="100%"><defs>
```

```
<filter id="myFilterGammaG" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%" height=
"100%"><feComponentTransfer><feFuncG type="gamma" amplitude="2" exponent="3" offset="0"/>
</feComponentTransfer></filter>
 <filter id="myFilterGammaR" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%"
height="100%"><feComponentTransfer><feFuncR type="gamma" amplitude="2" exponent="5" offset=
"0"/></feComponentTransfer></filter></defs>
 <image x="0" y="0" width="400" height="250" xlink:href="img/A211.jpg" /></svg>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feComponentTransfer 滤镜主要通过公式计算对图像的亮度、对比度等特性进行调整。feComponentTransfer 滤镜包含 4 个元素，它们分别是 feFuncR、feFuncG、feFuncB、feFuncA，也就是可以针对 R、G、B、A 进行独立调整，调整的类型(type)分别是 identity、table、discrete、linear、gamma。当前实例主要采用 gamma 方法调整图像。gamma 调整图像的公式为 C'＝amplitude * pow(C，exponent)＋offset，因此就具有 amplitude(振幅)、exponent(指数)、offset(偏移量)3 个参数。

此实例的源文件名是 myHtmlA212.html。

204　使用 SVG 滤镜以半个太极图裁剪图像

此实例主要通过使用 SVG 的 feComposite、feOffset 等滤镜将图像裁剪成半个太极图。当在 Google Chrome 浏览器中显示该页面时将显示默认创建的半个太极图，如图 204-1 所示；单击“使用半个太极图裁剪图像”按钮，则半个太极图的内部将被一幅图像填充，如图 204-2 所示。有关此实例的主要代码如下。

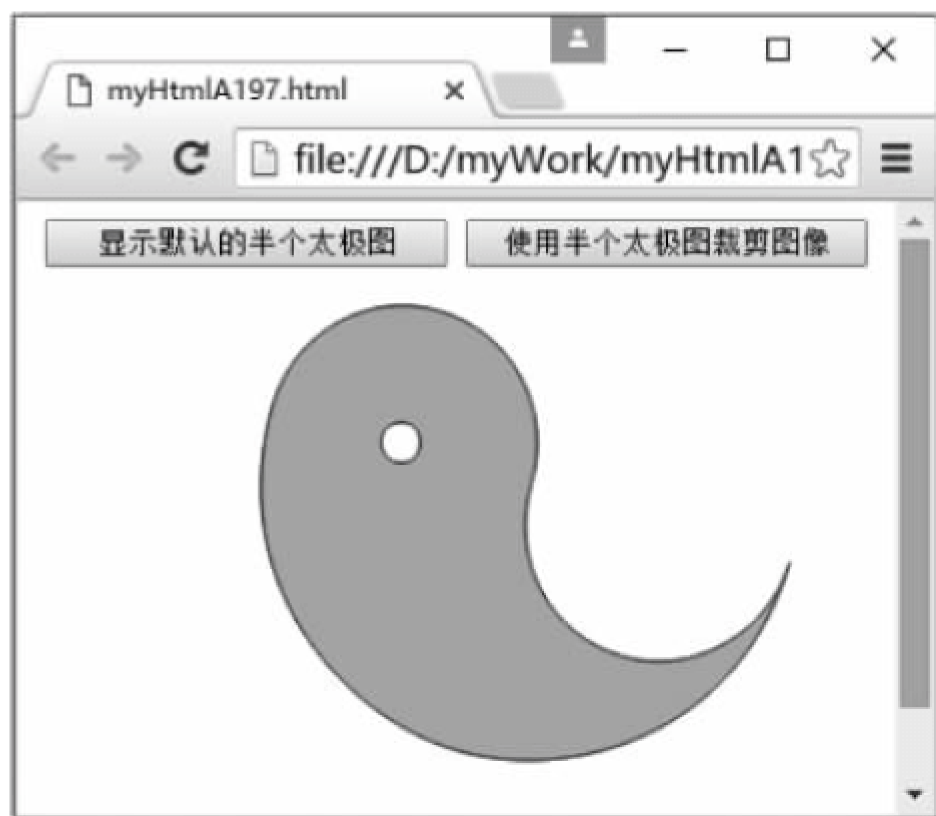


图　204-1



图　204-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtn1").click(function() {//显示默认的半个太极图
     $("path").attr("filter", "");
   });
    $("#myBtn2").click(function() {//使用半个太极图裁剪图像
     $("path").attr("filter", "url(#myFilter)");
   });});
```

```
</script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
  button { width: 175px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示默认的半个太极图</button>
 <button id="myBtn2">使用半个太极图裁剪图像</button></div>
<svg width="100%" height="100%">
 <defs><filter id="myFilter">
   <feImage xlink:href="img/A197.jpg" width="400" height="250" result="myA197" />
   <feComposite in="myA197" in2="SourceAlpha" operator="in"/></filter>
</defs>
 <path d="M193.1 79.292 C224.16 88.7968 242.127 121.499 232.627 154.414 C223.127 187.328 239.995
220.003 271.913 229.504 C303.83 239.006 336.35 218.61 345 188.876 C323.124 264.909 248.755 285.576
198.994 270.349 C149.231 255.12 98.1789 196.096 120.144 119.992 C129.376 88.0081 162.039 69.7869
193.1 79.292 zM167.341 136.242 C167.341 141.144 170.673 145.311 176.03 145.311 C181.386 145.311
184.242 140.899 184.242 136.242 C184.242 131.584 180.796 127.762 175.91 127.785 C171.025 127.809
167.341 131.339 167.341 136.242 z" transform="translate(0, -70)" style="fill: rgb(192,192,255);
stroke:rgb(130,0,61);stroke-width:1px;";/></svg> </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<path/>标签中的 d 属性值是绘制半个太极图的关键点坐标；<feComposite in="myA197" in2="SourceAlpha" operator="in"/>表示将半个太极图与 in="myA197"图像进行像素运算，operator 属性常用的属性值有 over、in、out、atop、xor、arithmetic，它们分别对应不同的算法，默认值为 over。

此实例的源文件名是 myHtmlA197.html。

205 使用 SVG 滤镜将图像裁剪成桃心图案

此实例主要通过使用 SVG 的 feComposite、feOffset 等滤镜将图像裁剪成桃心图案。当在 Google Chrome 浏览器中显示该页面时将显示创建的默认桃心图案，单击“使用桃心图案裁剪图像”按钮，则桃心图案的内部将被一幅图像填充，如图 205-1 所示。有关此实例的主要代码如下。



图 205-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示默认的桃心图案
    $("path").attr("filter", "");
   });
   $("#myBtn2").click(function() {//使用桃心图案裁剪图像
    $("path").attr("filter", "url(#myFilter)");
  });});
 </script>
 <style type="text/css">
  svg { margin-top: 10px; position: absolute; left: calc(50% - 200px);}
  button { width: 175px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示默认的桃心图案</button>
 <button id="myBtn2">使用桃心图案裁剪图像</button></div>
<svg width="100%" height="100%"><defs><filter id="myFilter">
   <feImage xlink:href="img/A198.jpg" width="400" height="250" result="myA198" />
   <feComposite in="myA198" in2="SourceAlpha" operator="in"/></filter></defs>
 <path d="M233.642 288 C249.9 265.728 318 231.266 318 195.613 C318 133.896 247.069 148.581 232.993
172.957 C218.041 147.832 149 137.805 149 195.613 C149 232.135 219.429 265.911 233.642 288 z" transform
 ="scale(1.95),translate(-130, -140)"
style="fill:red;stroke:black;stroke-width:1px;";/></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<path/>标签中的 d 属性值是绘制桃心图案的关键点坐标；<feComposite in="myA198" in2="SourceAlpha" operator="in"/>表示将桃心图案与 in="myA198"图像进行像素运算，即提取与桃心图案对应位置的图像像素。

此实例的源文件名是 myHtmlA198.html。

206 使用 SVG 路径将图像裁剪成任意形状

此实例主要通过使用 SVG 的 clip-path 属性将图像裁剪成任意形状。当在 Google Chrome 浏览器中显示该页面时将显示默认的原始图像，单击“显示裁剪的图像”按钮，图像经过预置的不规则路径裁剪之后的形状如图 206-1 所示。有关此实例的主要代码如下。



图 206-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示原始的图像
    $("image").attr("clip-path", ""); });
   $("#myBtn2").click(function() {//显示裁剪的图像
    $("image").attr("clip-path", "url(#myClip)"); });});
 </script>
 <style type="text/css">
  .mySVG { width: 410px; height: 410px;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示原始的图像</button>
 <button id="myBtn2">显示裁剪的图像</button></div>
<div align="center"><svg class="mySVG"><defs>
 <clipPath id="myClip" transform="scale(1.5,1),translate(-25,-50)">
  <path d="M66.039,133.545 c0,0-21-57,18-67 s49-4,65,8 s30,41,53,27 s66,4,58, 32 s-5,44,18,57
s22,46,0,45 s-54-40-68-16 s-40,88-83,48 s11-61-11-80 s-79-7 -70-41 C46.039,146.545,
53.039,128.545,66.039,133.545 z"/></clipPath></defs>
  <image x="-20" y="5" width="450" height="250" xlink:href="img/A219.jpg"/></svg>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<clipPath>标签用于隐藏位于剪切路径以外的对象部分，在<clipPath>标签中通常需要在子结点中使用<path>标签创建不规则路径。在其他标签中定义绘制什么和不绘制什么的模具被称为剪切路径(clip-path)，clip-path属性值指向的即是<clipPath>标签，例如 clip-path=" url(#myClip)"。

此实例的源文件名是 myHtmlA219.html。

207 在 SVG 中通过路径绘制多个扇形饼图

此实例主要通过使用 SVG 的<path>标签绘制多个扇形的饼图。当在 Google Chrome 浏览器中显示该页面时，通过路径绘制的多个扇形的饼图效果如图 207-1 所示。有关此实例的主要代码如下。

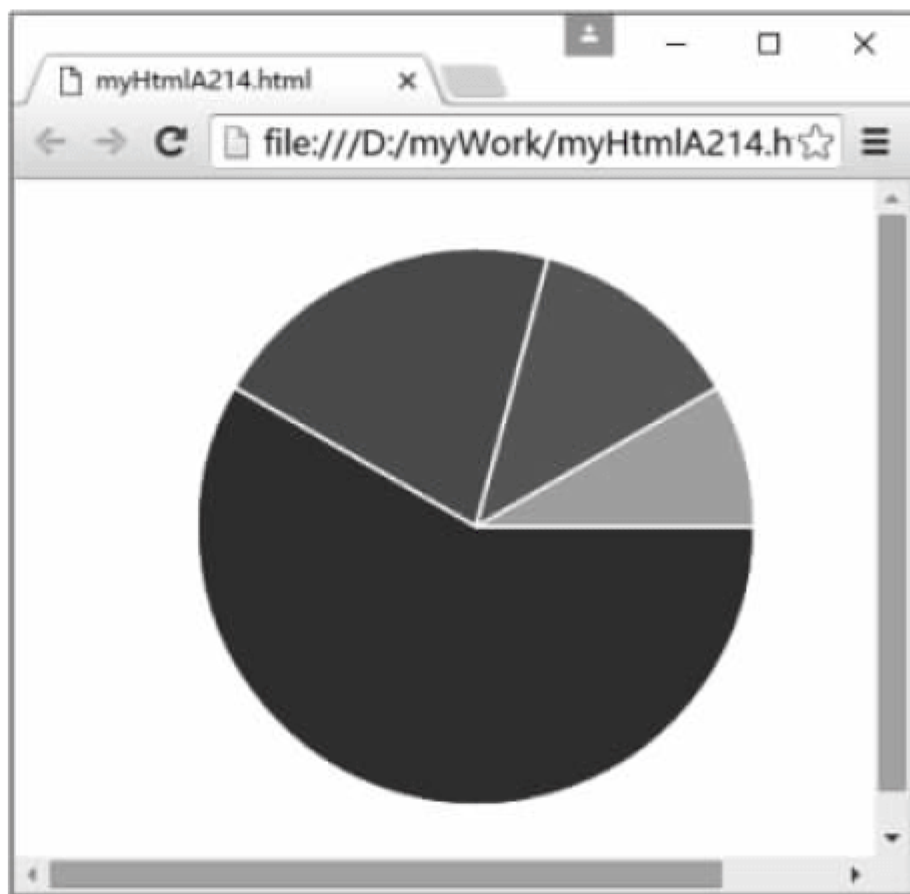


图 207-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var startAngle=0; var cx=150;
   var cy=150; var r=135;
   var myDeg1=30+startAngle;
   var myDeg2=45+myDeg1;
   var myDeg3=75+myDeg2;
   var myDeg4=210+myDeg3;
   var x0=cx+r*Math.cos(startAngle*Math.PI/180);
   var y0=cy-r*Math.sin(startAngle*Math.PI/180);
   var x1=cx+r*Math.cos(myDeg1*Math.PI/180);
   var y1=cy-r*Math.sin(myDeg1*Math.PI/180);
   var x2=cx+r*Math.cos(myDeg2*Math.PI/180);
   var y2=cy-r*Math.sin(myDeg2*Math.PI/180);
   var x3=cx+r*Math.cos(myDeg3*Math.PI/180);
   var y3=cy-r*Math.sin(myDeg3*Math.PI/180);
   var x4=cx+r*Math.cos(myDeg4*Math.PI/180);
   var y4=cy-r*Math.sin(myDeg4*Math.PI/180);
   $("#myPie1").attr("d", "M "+cx+","+cy+" L "+x0+","+y0+" A "+r+","+r+" 0 0,0 "+
x1+","+y1+" Z");
   $("#myPie2").attr("d", "M "+cx+","+cy+" L "+x1+","+y1+" A "+r+","+r+" 0 0,0 "+
x2+","+y2+" Z");
   $("#myPie3").attr("d", "M "+cx+","+cy+" L "+x2+","+y2+" A "+r+","+r+" 0 0,0 "+
x3+","+y3+" Z");
   //如果要绘制大于180°的饼图,路径角度弧线应为1,即"0 1,0"
   $("#myPie4").attr("d", "M "+cx+","+cy+" L "+x3+","+y3+" A "+r+","+r+" 0 1,0 "+
x4+","+y4+" Z");
  });
 </script>
 <style type="text/css">
  svg { margin-top: 10px;position: absolute;left: calc(50% - 135px);}
 </style></head>
<body><svg width="100%" height="100%">
 <path id="myPie1" style="fill:cyan; stroke:#FFF; stroke-width:2;"/>
 <path id="myPie2" style="fill:red; stroke:#FFF; stroke-width:2;"/>
 <path id="myPie3" style="fill:green; stroke:#FFF; stroke-width:2;"/>
 <path id="myPie4" style="fill:blue; stroke:#FFF; stroke-width:2;"/>
</svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,如果要绘制大于180°的饼图,路径角度弧线应为1,即"0 1,0",例如“$("#myPie4").attr("d", "M "+cx+","+cy+" L "+x3+","+y3+" A "+r+","+r+" 0 1,0 "+x4+","+y4+" Z")”,可以对比前面3行代码与此行代码的差异。

此实例的源文件名是myHtmlA214.html。

208　使用SVG滤镜实现仅显示图像的轮廓边缘

此实例主要通过在SVG中使用feSpecularLighting滤镜提取图像的轮廓边缘,以产生类似于版画的效果。当在Google Chrome浏览器中显示该页面时将显示原始的图像,单击“使用青色光源照射图像”按钮,提取的图像轮廓如图208-1所示;单击“使用黄色光源照射图像”按钮,提取的图像轮廓如

图 208-2 所示。有关此实例的主要代码如下。

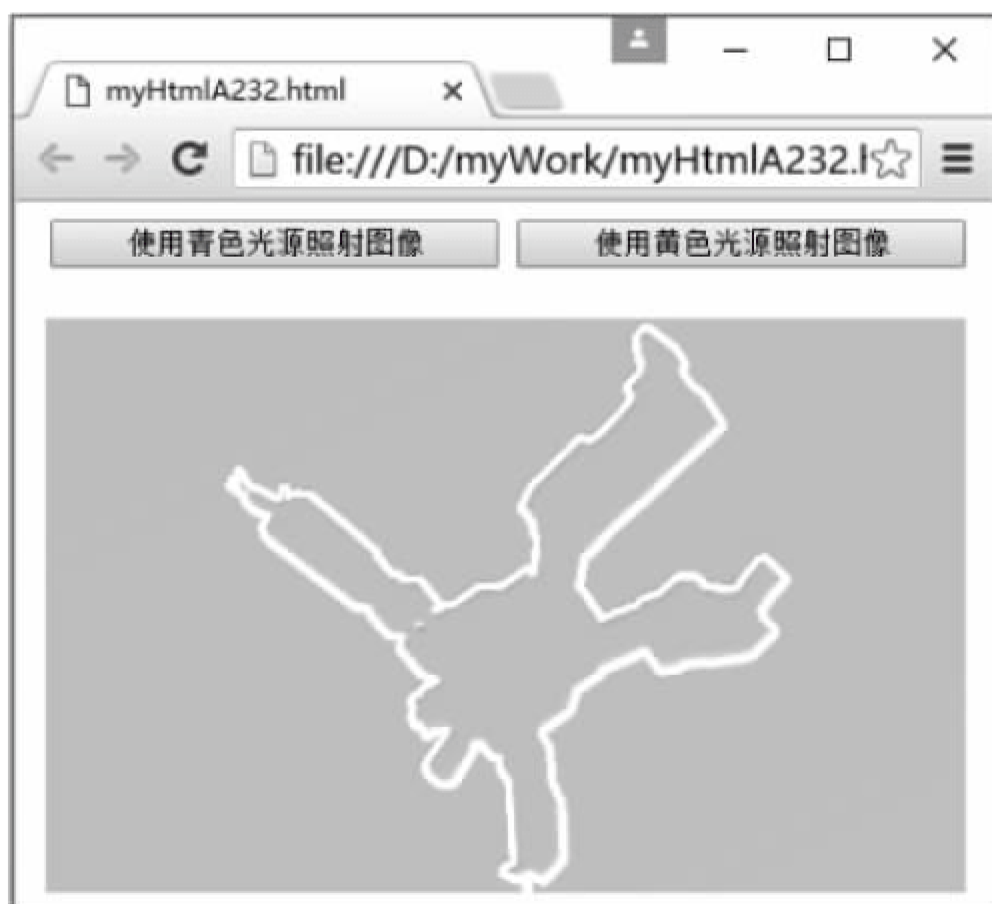


图 208-1

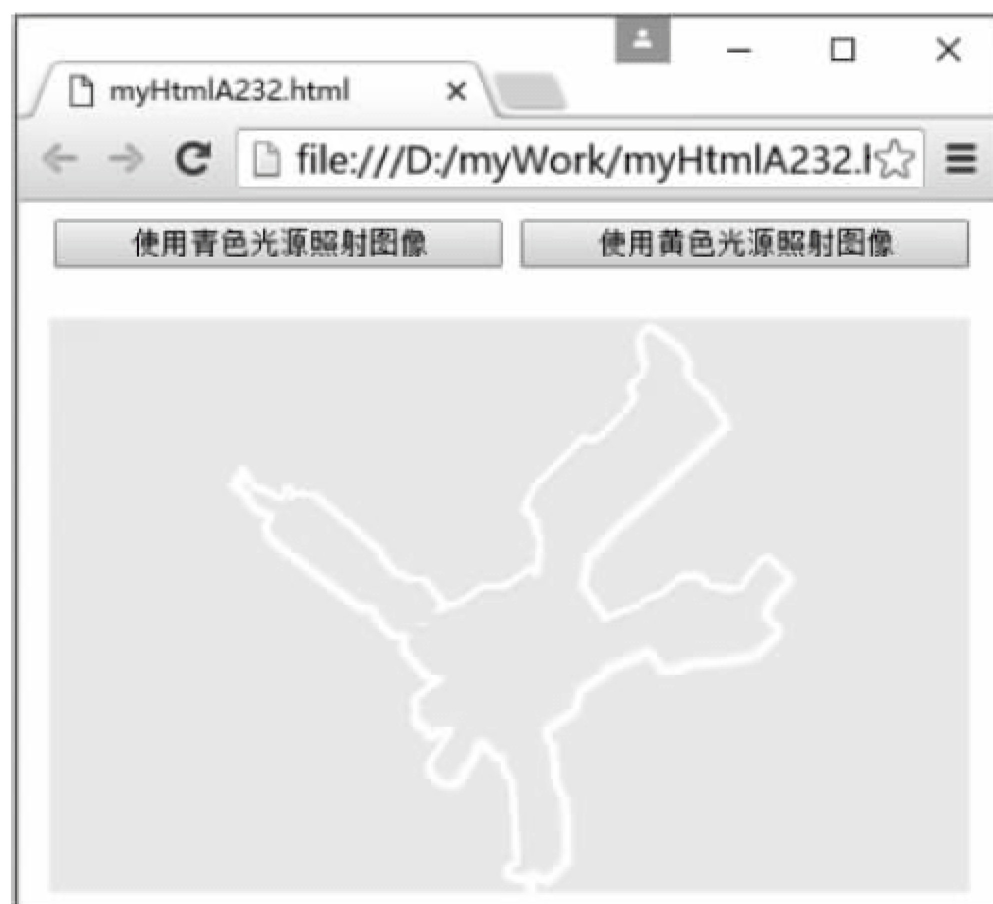


图 208-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//使用青色光源照射图像
    $("svg defs filter feSpecularLighting").attr("lighting-color", "cyan");
    $("g").attr("filter", "url(#myFilter)");
   });
   $("#myBtn2").click(function() {//使用黄色光源照射图像
    $("svg defs filter feSpecularLighting").attr("lighting-color", "yellow");
    $("g").attr("filter", "url(#myFilter)");
  }); });
 </script>
 <style type="text/css">
  svg { margin-top: 20px; position: absolute;left: calc(50% - 200px); }
  div { margin-top: 2px; margin-left: 2px;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">使用青色光源照射图像</button>
 <button id="myBtn2">使用黄色光源照射图像</button></div>
<div><svg width="400" height="250"><defs><filter id="myFilter">
  <feSpecularLighting in="SourceAlpha" surfaceScale="5" specularConstant="0.75"
specularExponent="20" lighting-color="yellow">
  <fePointLight x="-5000" y="-10000" z="20000"/>
 </feSpecularLighting></filter></defs>
 <g><image id="myImage" x="0" y="0" width="400px" height="250px" xlink:href="img/A195.png">
</image></g></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feSpecularLighting 滤镜的主要作用是采用标准的 Phong 照明模型(Phong lighting model)使图像根据光源产生凹凸的立体特效，在这种情况下 feSpecularLighting 滤镜通常与 feGaussianBlur 滤镜组合在一起使用，此实例采用独立的 feSpecularLighting 滤镜照射图像，从而使图像产生了版画轮廓的效果。

此实例的源文件名是 myHtmlA232.html。

209　使用 SVG 滤镜加粗或细化图形的轮廓

此实例主要在 SVG 中设置 feMorphology 滤镜的 operator 属性值，从而实现对图形图像的轮廓进行加粗或细化特效的处理。当在 Google Chrome 浏览器中显示该页面时将显示原始的蜘蛛图形，单击“对图形的轮廓进行细化处理”按钮，则图形轮廓经过 feMorphology 滤镜细化处理之后的效果如图 209-1 所示；单击“对图形的轮廓进行加粗处理”按钮，则图形轮廓经过 feMorphology 滤镜加粗处理之后的效果如图 209-2 所示。有关此实例的主要代码如下。



图　209-1



图　209-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//对图形的轮廓进行细化处理
    $("g").attr("filter", "url(#myErode)");
   });
   $("#myBtn2").click(function() {//对图形的轮廓进行加粗处理
    $("g").attr("filter", "url(#myDilate)");
  }); });
 </script>
 <style type="text/css">
  svg { margin-top: 20px; position: absolute;left: calc(50% - 200px);}
  div { margin-top: 2px; margin-left: 2px;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">对图形的轮廓进行细化处理</button>
 <button id="myBtn2">对图形的轮廓进行加粗处理</button></div>
<div><svg width="400" height="275">
 <defs>
   <filter id="myErode"> feMorphology operator="erode" in="SourceGraphic" radius="3.95" /></filter>
   <filter id="myDilate"><feMorphology operator="dilate" in="SourceGraphic" radius="3.9" />
</filter></defs>
```

```
  <g><image xlink:href = "img/A236.png" id = "MyImage" x = "0" y = "0" width = "400" height = "250" />
</g></svg></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feMorphology 滤镜用于对源图形执行 fattening(增肥)或者 thinning(瘦身)，当设置 feMorphology 滤镜的 operator 属性值为 erode 时该滤镜产生的是瘦身特效，当设置 feMorphology 滤镜的 operator 属性值为 dilate 时该滤镜产生的是增肥特效，瘦身和增肥特效的数值由 feMorphology 滤镜的 radius 属性值确定。除了图形图像的边框之外，此滤镜对图形图像中的轮廓也有较强的处理功能。

此实例的源文件名是 myHtmlA235.html。

210 使用 SVG 的 feTile 滤镜对图像进行平铺

此实例主要通过使用 SVG 的 feTile 滤镜实现在矩形中平铺图像。当在 Google Chrome 浏览器中显示该页面时将显示原始的图像，单击“显示平铺的图像”按钮，则图像在矩形中的平铺效果如图 210-1 所示。有关此实例的主要代码如下。



图 210-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
   $(function() {
    $("#myBtn1").click(function() {//显示原始的图像
     $("use").hide(); $("image").show();
    });
    $("#myBtn2").click(function() {//显示平铺的图像
     $("use").show();  $("image").hide();
    });
    $("#myBtn1").click();
});
 </script>
```

```
<style type="text/css">
  svg { margin-top: 20px; position: absolute; left: calc(50% - 200px); }
  div { margin-top: 2px; margin-left: 2px;}
  button { width: 195px;}
 </style></head>
<body><div align="center">
 <button id="myBtn1">显示原始的图像</button>
 <button id="myBtn2">显示平铺的图像</button></div>
<div><svg width="400" height="300"><defs>
   <filter id="MyFilter" filterUnits="userSpaceOnUse" x="0" y="0" width="100%" height="300">
<feImage x="0" y="0" width="100" height="100" xlink:href="img/A236.png" result='pict1'/><feTile
x="0" y="0" width="100%" height="300" in="pict1"/></filter></defs>
  <g><use filter="url(#MyFilter)" x='0' y='0'/><image xlink:href="img/A236.png" id="MyImage"
x="0" y="0" width="100" height="100" /></g></svg></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feTile滤镜用于将图像的多个副本平铺在指定的矩形范围内。当使用feTile滤镜平铺图像时通常需要指定in属性值代表的图像，指定width和height属性值代表的矩形范围，指定x和y属性值代表的矩形左上角坐标。

此实例的源文件名是myHtmlA236.html。

211　在SVG滤镜中使用不同的光源照射图像

此实例主要在SVG中使用feGaussianBlur、feSpecularLighting等滤镜，并在feSpecularLighting滤镜中使用不同颜色的光源，从而使图像显示不同的凹凸Phong立体特效。当在Google Chrome浏览器中显示该页面时，单击“使用青色光源照射图像”按钮，则图像产生的凹凸Phong立体特效如图211-1所示；单击“使用黄色光源照射图像”按钮，则图像产生的凹凸Phong立体特效如图211-2所示。有关此实例的主要代码如下。

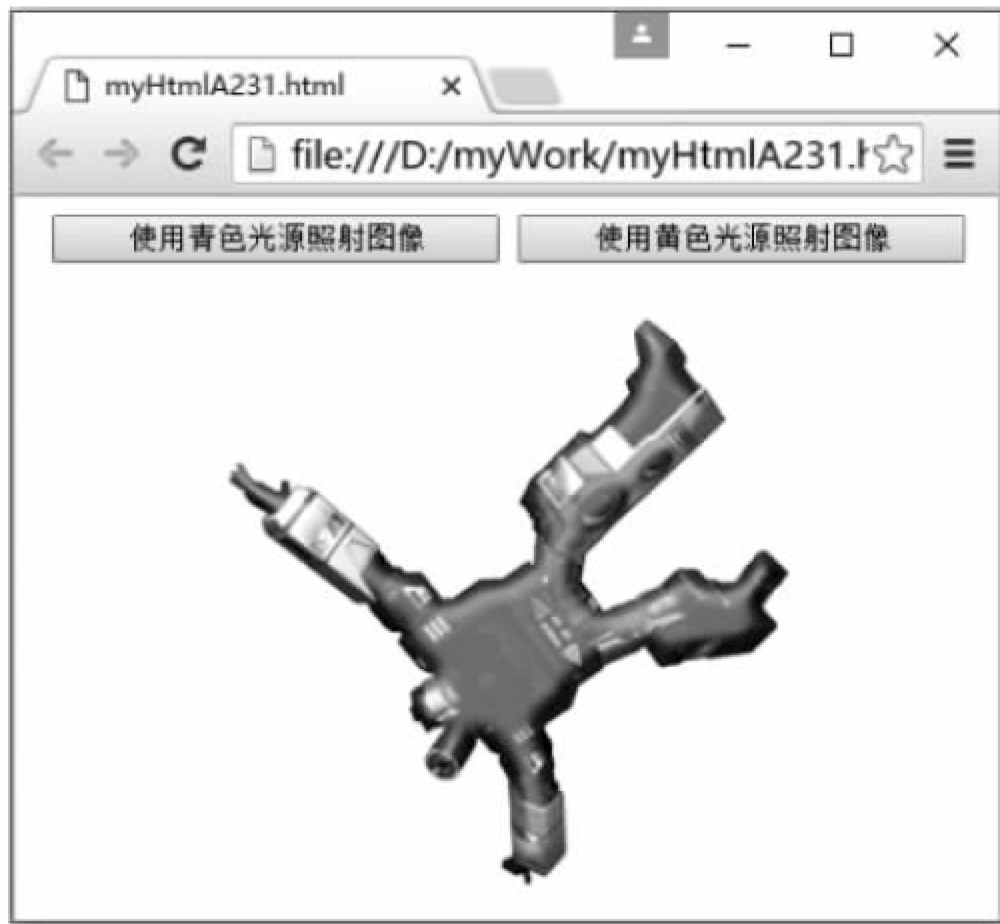


图　211-1

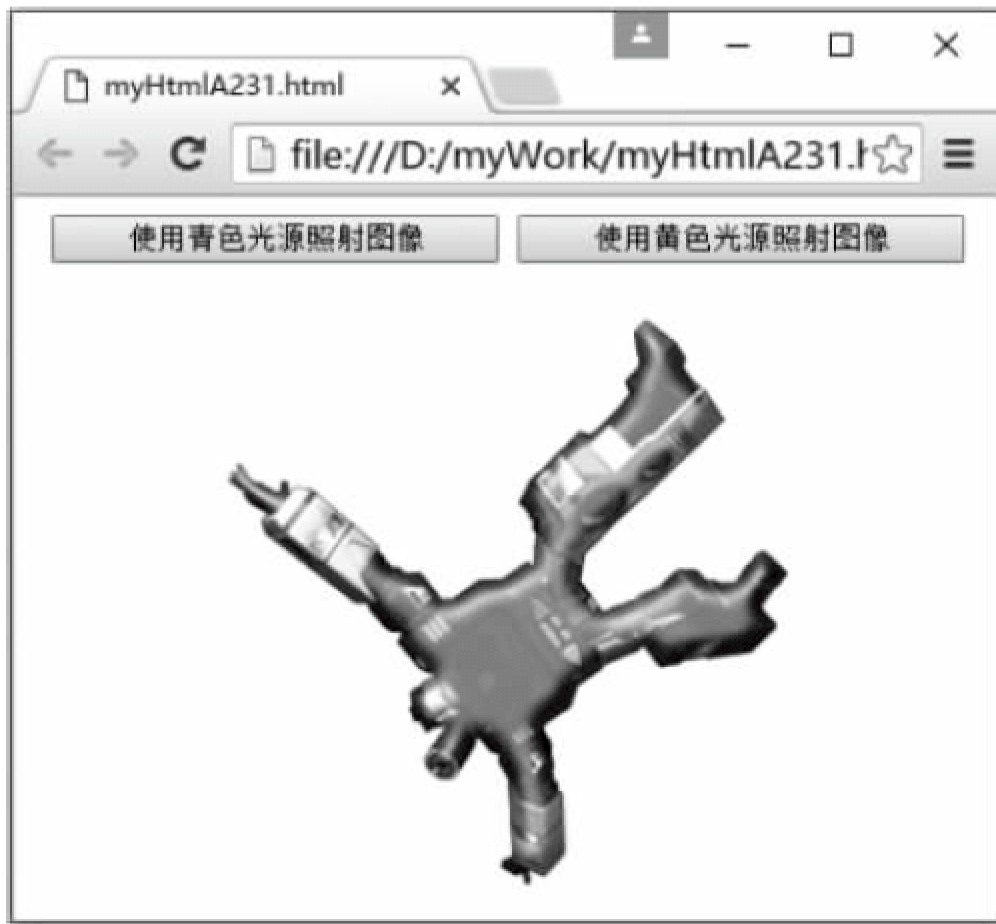


图　211-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
```

```
    $(function() {
     $("#myBtn1").click(function() {//使用青色光源照射图像
      $("svg defs filter feSpecularLighting").attr("lighting-color","cyan");
     });
     $("#myBtn2").click(function() {//使用黄色光源照射图像
      $("svg defs filter feSpecularLighting").attr("lighting-color","yellow");
    }); });
   </script>
   <style type="text/css">
    svg { margin-top: 20px;position: absolute; left: calc(50% - 200px); }
    div { margin-top: 2px;margin-left: 2px;}
    button{ width:195px;}
   </style></head>
  <body><div align="center">
   <button id="myBtn1">使用青色光源照射图像</button>
   <button id="myBtn2">使用黄色光源照射图像</button></div>
  <div><svg width="400" height="250"><defs><filter id="myFilter" >
       <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
       <feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
       <feSpecularLighting in="blur" surfaceScale="5" specularConstant="0.75"
  specularExponent="20" lighting-color="yellow" result="specOut">
        <fePointLight x="-5000" y="-10000" z="20000"/></feSpecularLighting>
     <feComposite in="specOut" in2="SourceAlpha" operator="atop" result="specOut"/>
     <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic" k1="0" k2="1" k3="1"
  k4="0" result="litPaint"/></filter></defs>
     <g filter="url(#myFilter)"><image id="myImage" x="0" y="0" width="400px" height="250px"
  xlink:href="img/A195.png"></image></g>
   </svg></div>
  </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feSpecularLighting 滤镜遵循标准的 Phong 照明模型(Phong lighting model)，使图像根据光源产生凹凸的立体特效。feSpecularLighting 滤镜通常与 feGaussianBlur 滤镜组合在一起实现这种特效，feGaussianBlur 滤镜产生的结果作为 feSpecularLighting 滤镜的输入。<fePointLight>标签内嵌于<feSpecularLighting>标签中，用于设置光源的位置，对最终的结果影响也较大。

此实例的源文件名是 myHtmlA231.html。

212 在 SVG 中合并使用滤镜创建的多个图层

此实例主要通过使用 SVG 的 feMerge 滤镜实现对滤镜创建的多个图层进行合并。当在 Google Chrome 浏览器中显示该页面时，单击“显示原始的文字”按钮，将显示原始的文字；单击“显示图层合并的文字”按钮，将把多个滤镜创建的图层合并在一起，效果如图 212-1 所示。有关此实例的主要代码如下。

```
  <!doctype html><html><head><meta charset="UTF-8">
   <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
    $(function() {
     $("#myBtn1").click(function() {//显示原始的文字
      $("#myView").attr("filter","");
```

```
      });
      $("#myBtn2").click(function() {//显示图层合并的文字
        $("#myView").attr("filter","url(#MyFilter)");
      });});
    </script>
    <style type="text/css">
     svg { margin-top: 10px; position: absolute;left: calc(50% - 200px);}
     button { width: 190px;}
    </style></head>
   <body><div align="center">
    <button id="myBtn1">显示原始的文字</button>
    <button id="myBtn2">显示图层合并的文字</button></div>
   <svg width="100%" height="100%"><defs>
     <filter id="MyFilter" filterUnits="userSpaceOnUse">
      <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
      <feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
      <feSpecularLighting in="blur" surfaceScale="5" specularConstant="0.75"
   specularExponent="20" lighting-color="#BBBBBB" result="specOut">
        <fePointLight x="-5000" y="-10000" z="20000"/></feSpecularLighting>
      <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut"/>
      <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic" k1="0" k2="1" k3="1" k4=
   "0" result="litPaint"/>
      <feMerge><feMergeNode in="offsetBlur"/><feMergeNode in="litPaint"/></feMerge>
   </filter></defs>
     <g id="myView" filter="url(#MyFilter)" >
      <path fill="none" stroke="#D90000" stroke-width="10" d="M50,190 C0,190 0,30 50,30 L350,30
   C400,30 400,190 350,190 z" />
      <path fill="#D90000" d="M60,180 C13,180 13,40 60,40 L340,40 C390,40 385,180 340,180 z" />
      <g fill="#FFFFFF" stroke="black" font-size="50" font-family="Verdana" ><text x="52" y=
   "126" >炫酷实例集锦</text></g></g></svg></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feMerge 滤镜用于合并图层，图层在 <feMerge>标签的子结点<feMergeNode>中。实例的图层合并过程如下。

(1) feGaussianBlur 接收输入的 SourceAlpha，这是源图形的 alpha 通道，结果被存储在名为 blur 的临时缓冲区中。



图 212-1

(2) feOffset 接收缓冲区 blur,并以 4 个像素为单位在正的 X 和 Y 方向中转换结果,然后创建一个名为 offsetBlur 的新缓冲区,此缓冲区稍后将被用于创建投影效果。

(3) feSpecularLighting 使用缓冲区 blur 作为表面提升的模型并生成来自单个点光源的光照效果,结果存储在缓冲区 specOut 中。

(4) feComposite 使用原始的源图形的 alpha 通道来遮盖步骤(3)的结果,以便使中间结果不会大于原始的源图形。缓冲区 specOut 被再次用于存储结果。

(5) feComposite 将反射光效果与原始的源图形结合在一起,结果被存放到缓冲区 litPaint 中。

(6) feMerge 将两个图层结合在一起。下面的图层由步骤(2)的投影效果构成,上面的图层由步骤(5)的反射光效果构成。

此实例的源文件名是 myHtmlA215.html。

213 使用 SVG 的 feTurbulence 滤镜创建图像

此实例主要通过使用 SVG 的 feTurbulence 滤镜实现根据噪声创建图像。当在 Google Chrome 浏览器中显示该页面时,单击"显示低噪声的图像"按钮,则使用 feTurbulence 滤镜创建的低噪声图像的效果如图 213-1 所示;单击"显示高噪声的图像"按钮,则使用 feTurbulence 滤镜创建的高噪声图像的效果如图 213-2 所示。有关此实例的主要代码如下。

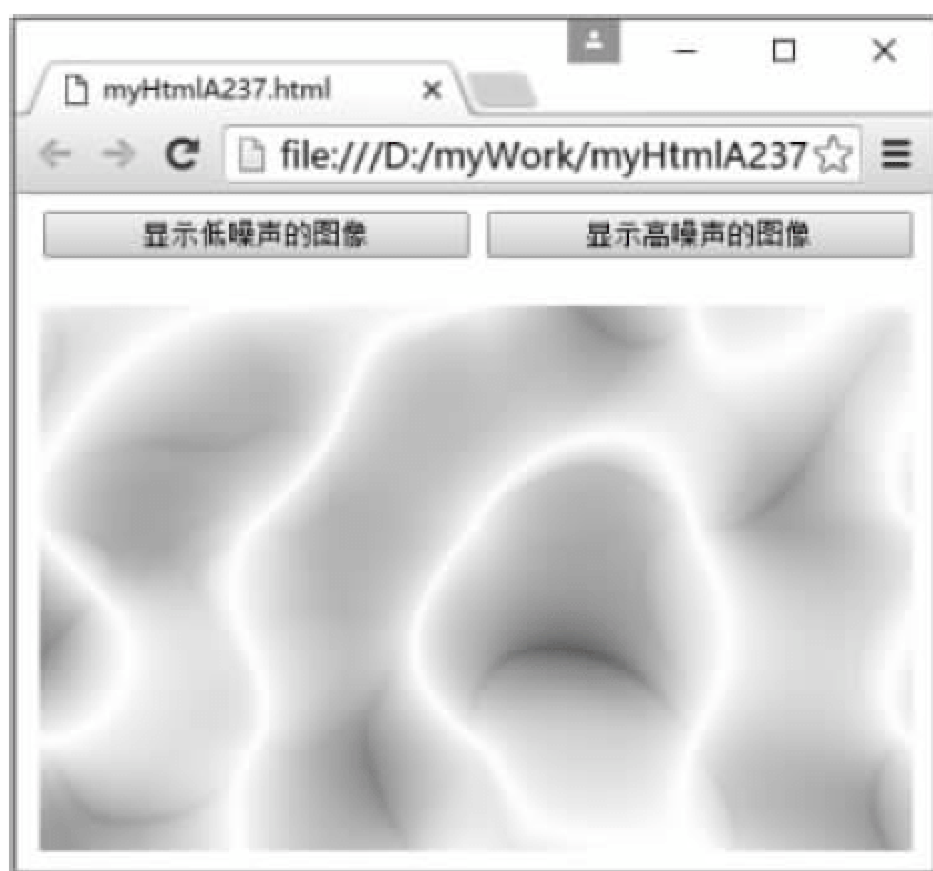


图 213-1

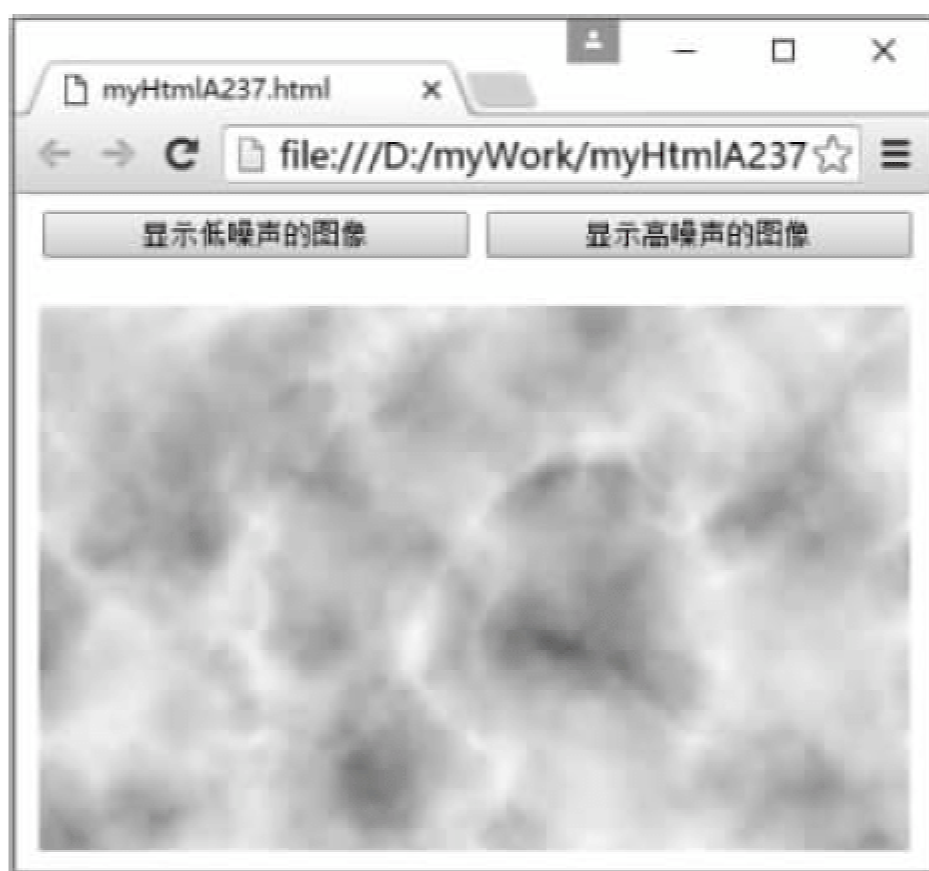


图 213-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtn1").click(function() {//显示低噪声的图像
    $("svg defs filter feTurbulence").attr("numOctaves", "1"); });
   $("#myBtn2").click(function() {//显示高噪声的图像
    $("svg defs filter feTurbulence").attr("numOctaves", "30"); });});
 </script>
 <style type="text/css">
  svg {margin-top: 20px;position: absolute;left: calc(50% - 200px);}
  div {margin-top: 2px;margin-left: 2px;}
  button { width: 195px;}
 </style></head>
```

```
<body><div align="center"><button id="myBtn1">显示低噪声的图像</button>
 <button id="myBtn2">显示高噪声的图像</button></div>
<div><svg width="400" height="250"><defs>
   <filter id="MyFilter" filterUnits="userSpaceOnUse" x="0" y="0" width="400" height="250">
<feTurbulence baseFrequency="0.01" numOctaves="1"/></filter></defs>
  <g><image id="myImage" x="0" y="0" width="400px" height="250px" xlink:href="img/A124.jpg"
filter="url(#MyFilter)"></image></g></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，feTurbulence 滤镜基于柏林扰动函数(Perlin turbulence function)创建图像，通过设置 feTurbulence 滤镜的 type、baseFrequency、numOctaves 等属性值就可以使图像实现不同的效果。例如，当设置 type 属性值为 fractalNoise 时(<feTurbulence type="fractalNoise" baseFrequency="0.01" numOctaves="1"/>)，创建的图像就比较模糊；当设置 type 属性值为 turbulence 时(<feTurbulence type="turbulence" baseFrequency="0.01" numOctaves="1"/>)，创建的图像就比较细腻。type 属性的默认值为 turbulence。

此实例的源文件名是 myHtmlA237.html。

214 在一个元素中设置线性渐变背景颜色

此实例主要通过在 CSS 样式中设置元素的 background 属性为 linear-gradient 使元素(div)的背景呈现线性渐变的颜色。当在 Google Chrome 浏览器中显示该页面时，由蓝向绿渐变的背景效果如图 214-1 所示。有关此实例的主要代码如下。

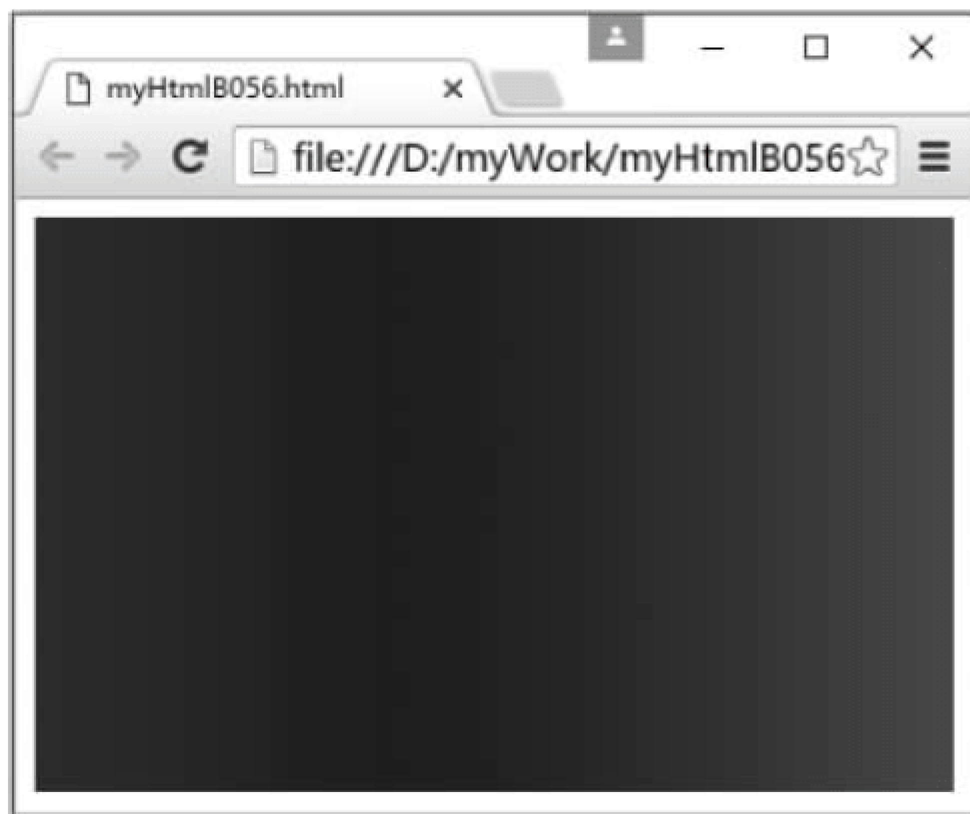


图 214-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {background: -webkit-linear-gradient(left, blue, green);
       width:400px;height: 250px;} /* 沿着水平方向从左至右由蓝到绿渐变 */
 </style>
<body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性用于一次性设置所有与背景相关的属性，可以在此属性中设置 background-attachment、background-position、background-color、background-size、background-repeat、background-origin、background-clip、background-image

等，如果不设置其中的某个值，也不会出问题，这些值的意义如表 214-1 所示。

表 214-1　在 background 属性中可以设置的值

值	描　　述
background-color	规定要使用的背景颜色
background-position	规定背景图像的位置
background-size	规定背景图像的尺寸
background-repeat	规定如何重复背景图像
background-origin	规定背景图像的定位区域
background-clip	规定背景的绘制区域
background-attachment	规定背景图像是否固定或者随着页面的其余部分滚动
background-image	规定要使用的背景图像
inherit	规定应该从父元素继承 background 属性的设置

渐变(gradient)是在两个或多个指定的颜色之间显示平稳的过渡，渐变的元素在放大时看起来效果更好，因为渐变是由浏览器生成的。linear-gradient 的语法格式如下。

```
linear - gradient(direction, color - stop1, color - stop2, … );
```

其中，direction 默认为 to bottom，即从上向下的渐变；stop 表示颜色的分布位置，默认均匀分布，如果有 3 个颜色，则各个颜色的 stop 均为 33.33%。

此实例的源文件名是 myHtmlB056.html。

215　在一个元素中叠加显示多个背景图像

此实例主要通过在 CSS 样式中设置元素的 background-image、background-position、background-repeat 等属性值实现在一个元素(div)中叠加显示多个背景图像。当在 Google Chrome 浏览器中显示该页面时，最底层的背景图像不重复显示，中间层的背景图像纵向重复显示，最上层的背景图像横向重复显示，如图 215-1 所示。有关此实例的主要代码如下。

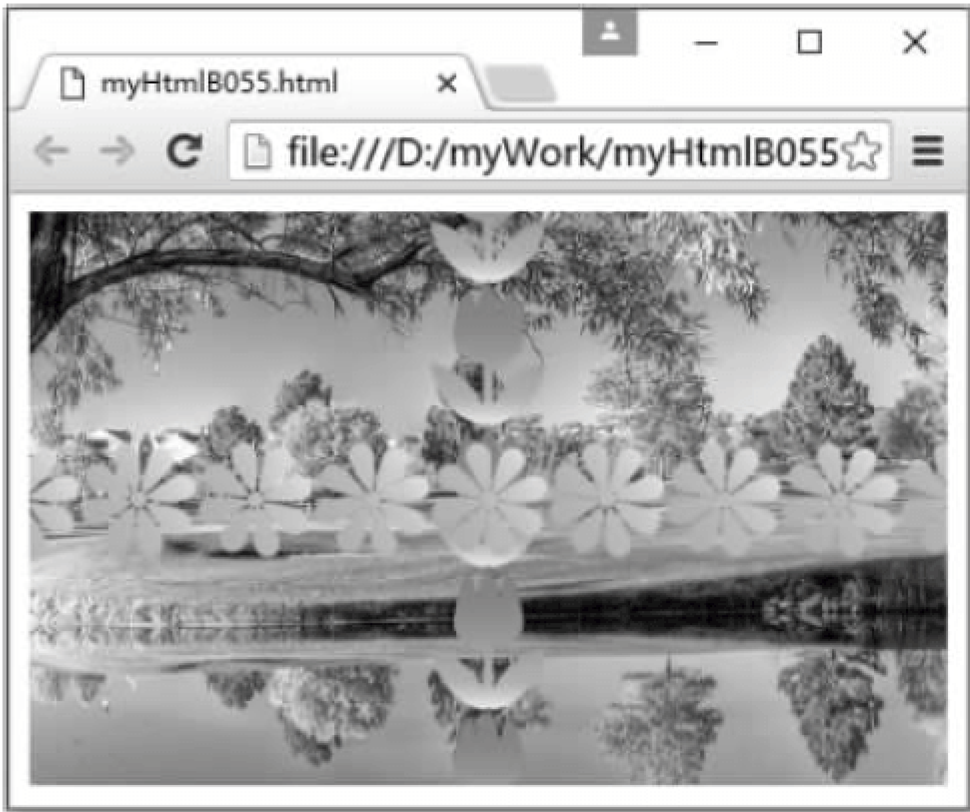


图　215-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
```

```
div{background - image: url(img/B055A.png),url(img/B055B.png), url(img/a048.jpg); background - repeat: repeat - x, repeat - y, no - repeat; background - position: center, center, center;width: 400px; height:250px;}
 </style>
<body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-image 属性用于为元素设置背景图像，元素的背景占据了元素的全部尺寸，包括内边距和边框，但不包括外边距；background-repeat 属性用于定义图像的平铺模式；background-position 属性用于设置背景图像的起始位置，这个属性设置背景原图像（由 background-image 定义）的位置，背景图像如果要重复，将从这一点开始。background-position 属性的取值和说明如表 215-1 所示。

表 215-1　background-position 属性的取值和说明

值	说　　明
top left、top center、top right、center left、center center、center right、bottom left、bottom center、bottom right	如果仅规定了一个关键字，那么第二个值将是 center，默认值是 0% 0%
x% y%	第一个值是水平位置，第二个值是垂直位置；左上角是 0% 0%，右下角是 100% 100%；如果仅规定了一个值，另一个值将是 50%
xpos ypos	第一个值是水平位置，第二个值是垂直位置；左上角是 0 0，单位是像素(0px 0px)或任何其他的 CSS 单位；如果仅规定了一个值，另一个值将是 50%，可以混合使用%和 position 值

此实例的源文件名是 myHtmlB055.html。

216　在一个元素中平铺显示多个背景图像

此实例主要在元素的 background-image 属性中使用逗号设置多个 url，从而实现多个不同的背景图像同时平铺显示。当在 Google Chrome 浏览器中显示该页面时将在垂直方向上显示 3 幅不同的背景图像，如图 216-1 所示。有关此实例的主要代码如下。



图　216-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { width: 450px; height:390px;
        background:url(img/B125A.jpg) no-repeat top right, url(img/B125B.jpg) repeat-x 0px 130px,
url(img/B125C.jpg) repeat-x bottom; }
 </style></head>
<body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性可以设置与元素背景相关的所有属性，例如 background-size、background-repeat、background-color、background-position、background-origin、background-clip、background-attachment、background-image。当需要在元素里应用多个背景图像时，引用图像之间需要用逗号“，”隔开。

此实例的源文件名是 myHtmlB125.html。

217 在元素中同时设置背景图像和背景颜色

此实例主要在 background 属性中同时设置 background-color、background-image 和 background-repeat 等属性值，从而实现在元素中同时显示背景图像和背景颜色。当在 Google Chrome 浏览器中显示该页面时，显示的海绿色为背景颜色，星形花瓣为背景图像，如图 217-1 所示。有关此实例的主要代码如下。



图 217-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body{ background: lightseagreen url(img/B055A.png) repeat ;}
 </style></head>
<body><p>同时设置背景图像和背景颜色</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性用于一次性设置所有与背景相关的属性，可以在此属性中设置 background-attachment、background-position、background-color、background-size、background-repeat、background-origin、background-clip、background-image 等。

此实例的源文件名是 myHtmlB057.html。

218　在元素背景之上叠加渐变色的遮罩层

此实例主要通过设置元素的 mask 属性实现在元素的背景之上叠加渐变色的遮罩层。当在 Google Chrome 浏览器中显示该页面时，单击“从左向右渐变”按钮，则背景和渐变色叠加的效果如图 218-1 所示；单击“从右向左渐变”按钮，则背景和渐变色叠加的效果如图 218-2 所示。有关此实例的主要代码如下。



图　218-1

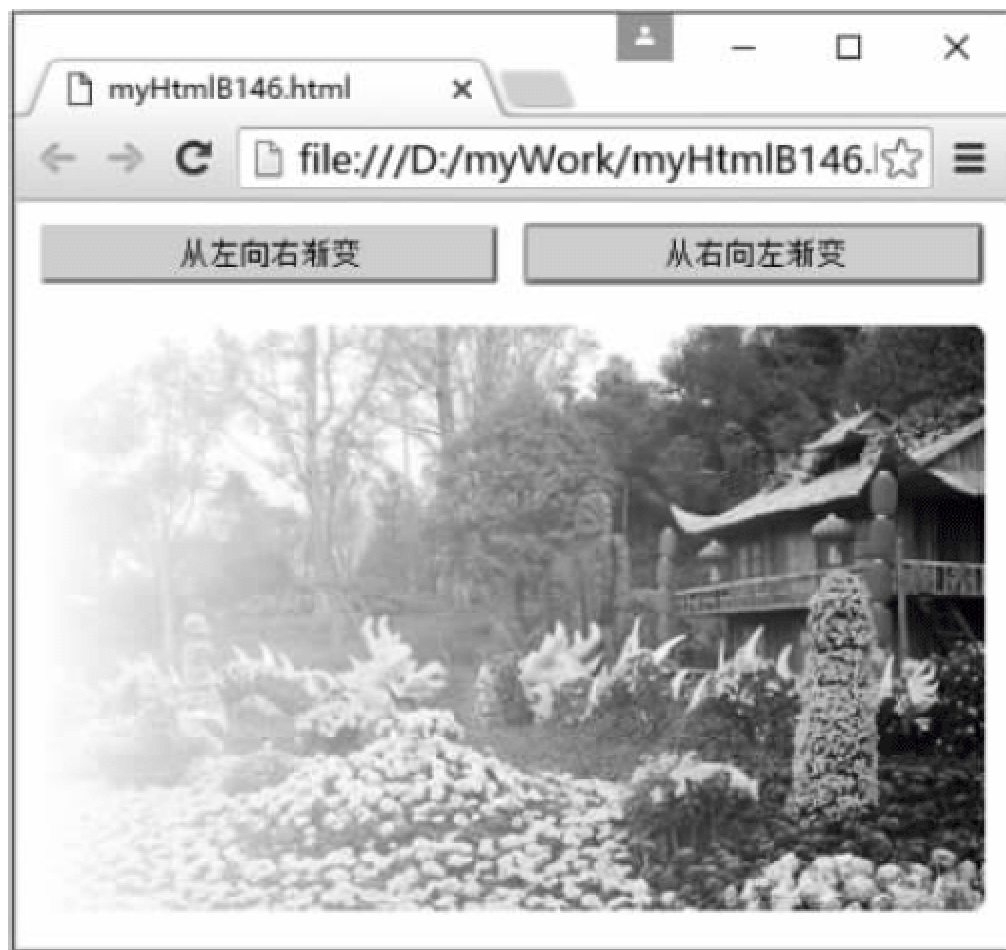


图　218-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnLTR").click(function() {//从左向右渐变
      $("p").css("-webkit-mask", "-webkit-gradient(linear, 0% 0%, 100% 0%, from(rgba(0,0,0,
1)), to(rgba(0,0,0,0)))");
    });
    $("#myBtnRTL").click(function() {//从右向左渐变
      $("p").css("-webkit-mask", "-webkit-gradient(linear, right top, left top, from(rgba(0,0,0,
1)), to(rgba(0,0,0,0)))");
   });});
 </script>
 <style type="text/css">
  p { background: url(img/B114.jpg) repeat;
      width:405px; height:250px; border-radius: 5px; }
  input {width: 195px;border-radius: 2px; padding: 3px;margin: 2px;}
 </style></head>
<body><div><input type="button" value="从左向右渐变" id="myBtnLTR"/>
 <input type="button" value="从右向左渐变" id="myBtnRTL"/></div><p></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background：url(img/B114.jpg) repeat 用于设置元素的背景图像；$("p").css("-webkit-mask"，"-webkit-gradient(linear，right top，left top，from(rgba(0,0,0,1))，to(rgba(0,0,0，0)))")用于将渐变背景叠加到背景图像上。如

果设置元素的 background：-webkit- gradient(linear, left top, right top, from(rgba(0,0,0,0.8)), to(rgba(0,0,0, 0.2)))，可以直接看到这个遮罩层实际上是一个由黑色到白色的渐变图。-webkit-gradient 用于创建渐变遮罩层，其语法格式如下。

```
webkit-gradient(type,x1 y1, x2 y2, from(开始颜色值),to(结束颜色值), [color-stop(偏移量<小数>,停靠颜色值), … ]);
```

webkit 内核的 Linear Gradients（线性渐变）的第一组参数 type(类型)为 linear。第二组参数是 x1 y1 和 x2 y2，即渐变色的两个点的坐标，x1、x2、y1、y2 的取值范围为 0%～100%，x1 和 x2 可以取值 left(或 0%)或 right(或 100%)，y1 和 y2 可以取值 top(或 0%)或 bottom(或 100%)，当 x1 等于 x2、y1 不等于 y2 时实现垂直渐变，调整 y1、y2 的值可以调整渐变半径的大小；当 y1 等于 y2、x1 不等于 x2 时实现水平渐变，调整 x1、x2 的值可以调整渐变半径的大小；当 y1 不等于 y2、x1 不等于 x2 时实现角度渐变，当 x1、x2、y1、y2 的取值为极值的时候接近垂直渐变或水平渐变；当 x1 等于 x2、y1 等于 y2 时没有渐变，取 from 颜色值；在实现垂直渐变和水平渐变时，x1 和 x2 可以直接取值 left(或 0%)或 right(或 100%)，y1 和 y2 可以直接取值 top(或 0%)或 bottom(或 100%)。from(开始颜色值)和 to(结束颜色值)是两个渐变颜色值。对于[color-stop(偏移量<小数>,停靠颜色值),…]，可以使用多个 color-stop，如果渐变只有两个颜色，那么可以不使用该参数；偏移量必须为小数，如果偏移量>=1，那么该 color-stop 相当于无效，此实例暂不考虑 color-stop。

此实例的源文件名是 myHtmlB146.html。

219 为元素设置从中心向圆周放射渐变的背景

此实例主要在 CSS 样式中设置元素的 background 属性为 radial-gradient，从而实现使元素(div)的背景呈现 3 种颜色顺次从中心向圆周放射渐变的效果。当在 Google Chrome 浏览器中显示该页面时，红、绿、蓝 3 色从中心向圆周放射渐变的背景效果如图 219-1 所示。有关此实例的主要代码如下。

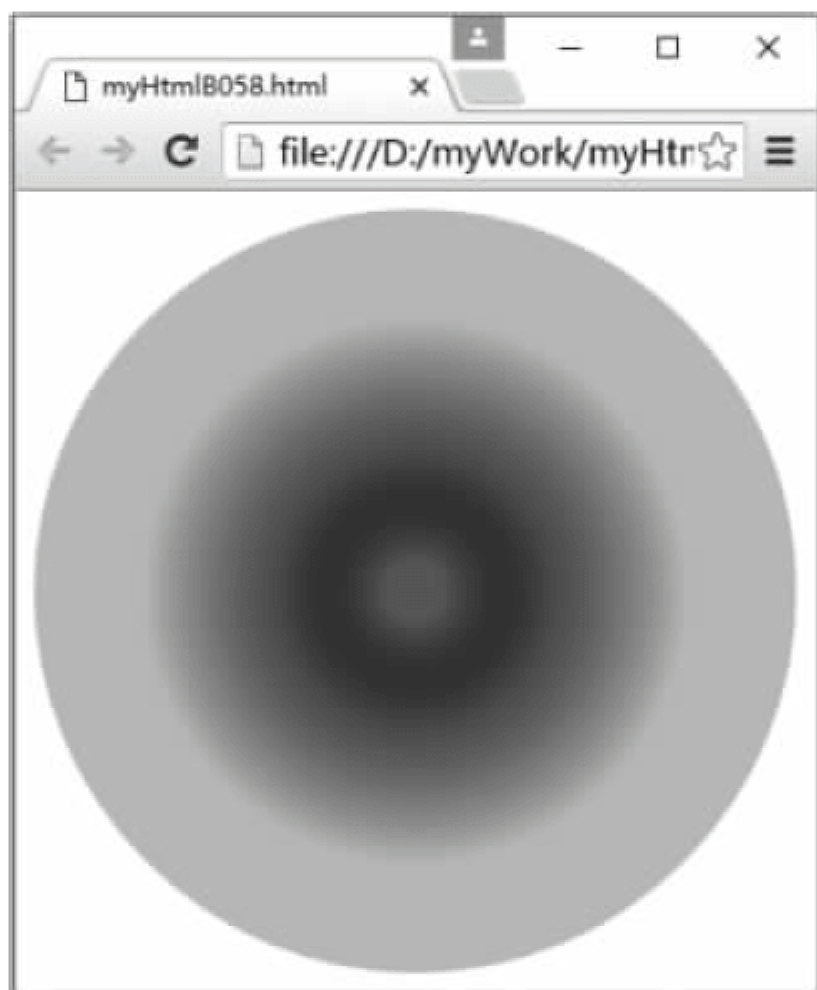


图 219-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
    div {width: 350px;height: 350px;border - radius: 175px;
         text - align: center; display: table - cell; vertical - align: middle;
         /* 按照 5%、25%、50% 的比例配置 3 种颜色的放射渐变过程 */
         background: - webkit - radial - gradient(red 5%, green 25%, lightblue 50%);}
   </style></head>
 <body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性用于一次性设置所有与背景相关的属性，如果其属性值是 radial-gradient，则会呈现中心放射渐变的背景。radial-gradient 的语法格式如下。

```
radial - gradient([ <position>,]? [ [ <shape> || <size> ] | <shape - size>{2},]? <color - stop>
[, <color - stop>] + )
```

各个参数的说明如下。

(1) <position>: [<length> | <percentage> | left | center | right]? [<length> | <percentage> | top | center | bottom]

其中，<length>表示用长度值指定径向渐变圆心的横坐标值，可以为负值；<percentage>表示用百分比指定径向渐变圆心的横坐标值，可以为负值；left 设置左边为径向渐变圆心的横坐标值；center 设置中间为径向渐变圆心的横坐标值；right 设置右边为径向渐变圆心的横坐标值；<length>用长度值指定径向渐变圆心的纵坐标值，可以为负值；<percentage>用百分比指定径向渐变圆心的纵坐标值，可以为负值；top 设置顶部为径向渐变圆心的纵坐标值；center 设置中间为径向渐变圆心的纵坐标值；bottom 设置底部为径向渐变圆心的纵坐标值。注意，同名参数按照出现顺序说明。

(2) <shape>: circle | ellipse

其中，circle 指定圆形的径向渐变；ellipse 指定椭圆形的径向渐变。

(3) <size>: closest-side | closest-corner | farthest-side | farthest-corner | contain | cover

其中，closest-side 指定径向渐变的半径长度为从圆心到离圆心最近的边；closest-corner 指定径向渐变的半径长度为从圆心到离圆心最近的角；farthest-side 指定径向渐变的半径长度为从圆心到离圆心最远的边；farthest-corner 指定径向渐变的半径长度为从圆心到离圆心最远的角；contain 指定径向渐变的半径长度为从圆心到离圆心最近的点，类同于 closest-side；cover 指定径向渐变的半径长度为从圆心到离圆心最远的点，类同于 farthest-corner。

(4) <shape-size>: <length> | <percentage>

其中，<length>用长度值指定径向渐变的横向或纵向直径长度，并根据横向和纵向的直径来确定是圆或椭圆，不允许为负值；<percentage>用百分比指定径向渐变的横向或纵向直径长度，并根据横向和纵向的直径来确定是圆或椭圆，不允许为负值。

(5) <color-stop>: <color> [<length> | <percentage>]

其中，<color>指定颜色；<length>用长度值指定起止色位置，不允许为负值；<percentage>用百分比指定起止色位置。

此实例的源文件名是 myHtmlB058.html。

220 为元素设置重复的、从中心放射渐变的背景

此实例主要在 CSS 样式中设置元素的 background 属性为 repeating-radial- gradient，从而实现使元素(div)的背景呈现 4 种颜色顺次从中心向圆周放射渐变，并且重复两次的效果。当在 Google

Chrome 浏览器中显示该页面时，黄、绿、红、白 4 色从中心向圆周放射渐变，并且重复两次（即黄、绿、红、白、黄、绿、红、白）的背景效果如图 220-1 所示。有关此实例的主要代码如下。

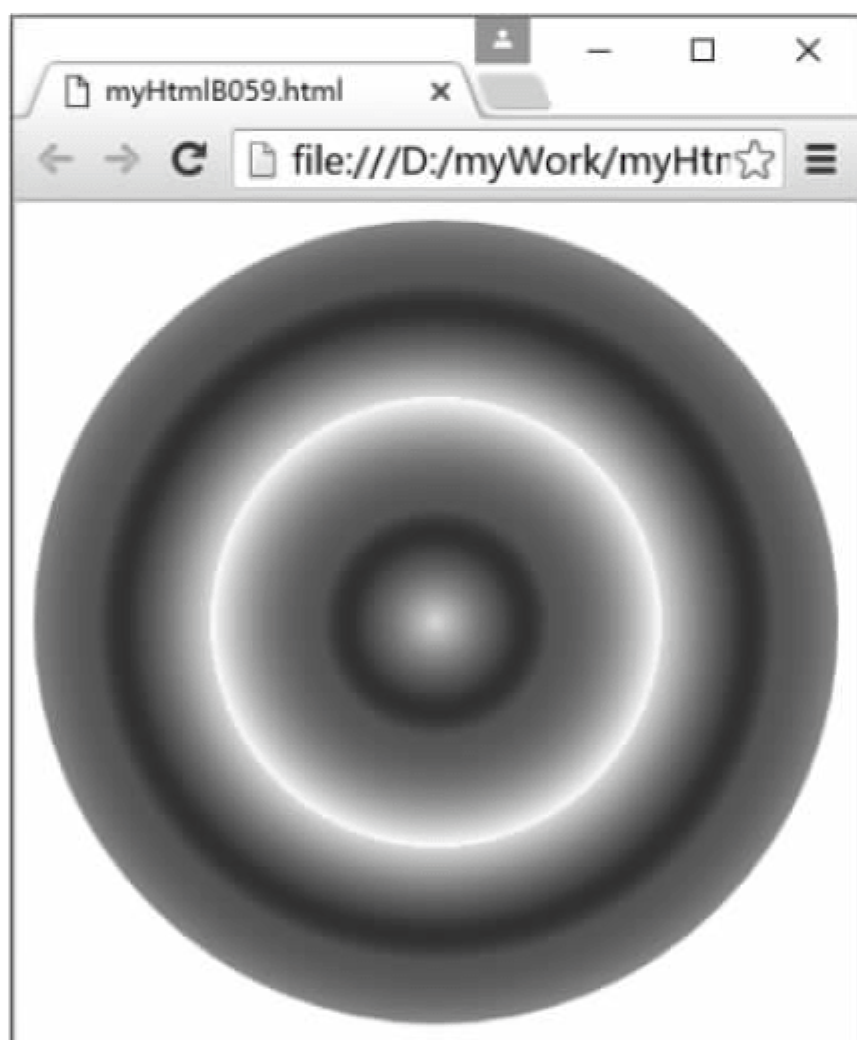


图　220-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { width: 350px; height: 350px; border-radius: 175px;
        text-align: center; display: table-cell;vertical-align: middle;
        background: -webkit-repeating-radial-gradient(yellow, green 30px, red 50px, white 98px);}
  </style></head>
<body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性用于一次性设置所有与背景相关的属性，如果其属性值是 repeating-radial-gradient，则会呈现可重复任意次数的中心放射渐变背景。repeating-radial-gradient 的语法格式如下。

```
repeating-radial-gradient([ <position>,]? [ [ <shape> || <size> ] | <shape-size>{2},]? <color-stop>[, <color-stop>] + )
```

各个参数的说明如下。

（1）<position>：[<length> | <percentage> | left | center | right]? [<length> | <percentage> | top | center | bottom]

其中，<length>表示用长度值指定径向渐变圆心的横坐标值，可以为负值；<percentage>表示用百分比指定径向渐变圆心的横坐标值，可以为负值；left 用于设置左边为径向渐变圆心的横坐标值；center 用于设置中间为径向渐变圆心的横坐标值；right 用于设置右边为径向渐变圆心的横坐标值；<length>表示用长度值指定径向渐变圆心的纵坐标值，可以为负值；<percentage>表示用百分比指定径向渐变圆心的纵坐标值，可以为负值；top 用于设置顶部为径向渐变圆心的纵坐标值；center 用于设置中间为径向渐变圆心的纵坐标值；bottom 用于设置底部为径向渐变圆心的纵坐标值。注意，同名参数按照出现顺序说明。

（2）<shape>：circle | ellipse

其中，circle 指定圆形的径向渐变；ellipse 指定椭圆形的径向渐变。

(3) < size >: closest-side | closest-corner | farthest-side | farthest-corner | contain | cover

其中,closest-side 指定径向渐变的半径长度为从圆心到离圆心最近的边;closest-corner 指定径向渐变的半径长度为从圆心到离圆心最近的角;farthest-side 指定径向渐变的半径长度为从圆心到离圆心最远的边;farthest-corner 指定径向渐变的半径长度为从圆心到离圆心最远的角;contain 指定径向渐变的半径长度为从圆心到离圆心最近的点,类同于 closest-side;cover 指定径向渐变的半径长度为从圆心到离圆心最远的点,类同于 farthest-corner。

(4) < shape-size >: < length > | < percentage >

其中,< percentage >表示用百分比指定径向渐变的横向或纵向直径长度,并根据横向和纵向的直径来确定是圆或椭圆,不允许为负值;< length >表示用长度值指定径向渐变的横向或纵向直径长度,并根据横向和纵向的直径来确定是圆或椭圆,不允许为负值。

(5) < color-stop >=< color > [< length > | < percentage >]

其中,< color >指定颜色;< length >表示用长度值指定起止色位置,不允许为负值;< percentage>表示用百分比指定起止色位置。

此实例的源文件名是 myHtmlB059. html。

221　以左上角为中心设置放射渐变背景

此实例主要在 CSS 样式中设置元素的 background 属性为 radial-gradient,从而使元素(div)的背景呈现 3 种颜色顺次以左上角为中心向四周放射渐变的效果。当在 Google Chrome 浏览器中显示该页面时,红、黄、白 3 色以左上角为中心向四周放射渐变的效果如图 221-1 所示。有关此实例的主要代码如下。



图　221-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  div {display: inline - block;width: 200px;height: 218px;margin: 10px;
       background - color: #FFF;border: 1px solid #EEE; position: relative;
       box - shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset;
       border - radius: 5px;
       background: radial - gradient(at top left, red 0 % ,yellow 50 % ,white 100 % );}
   div:first - letter {font - size: 24px; /* 设置首字样式 */
background - color: white; border - radius: 5px;box - shadow: 0px 0px 10px 5px gray;}
```

```
    p { padding - left: 15px; padding - right: 10px; font - size: 14px;}
    body { background - color: lightgray;}
  </style></head>
<body><div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div>
<div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background 属性用于一次性设置所有与背景相关的属性,如果其属性值是 radial-gradient,则将呈现中心放射渐变的背景。radial-gradient 的语法格式如下。

```
radial - gradient([ <position>,]? [ [ <shape> || <size> ] | <shape - size>{2},]? <color - stop>
[, <color - stop>] + )
```

在此实例中,radial-gradient(at top left, red 0%,yellow 50%,white 100%)中的 at top left 表示放射中心是左上角,以中心为起点,按照所示比例渐变的颜色依次是 red、yellow、white。

此实例的源文件名是 myHtmlB061.html。

222　以左下角为中心设置放射渐变背景

此实例主要在 CSS 样式中设置元素的 background 属性为 repeating-radial- gradient,从而实现使元素(div)的背景呈现 4 种颜色顺次以左下角为中心向四周放射渐变、重复数次的效果。当在 Google Chrome 浏览器中显示该页面时,yellow、lightblue、lightgreen、lightcyan 几色以左下角为中心向四周重复放射渐变的效果如图 222-1 所示。有关此实例的主要代码如下。



图　222-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  div {display: inline - block; width: 200px; height: 218px;border: 1px solid #EEE;
    margin: 10px; background - color: #FFF; position: relative; border - radius: 5px;
```

```
    box - shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset;
    background: repeating - radial - gradient (at bottom left, yellow, lightblue 2 %, lightgreen 3 %,
lightcyan 15 %);}
  div:first - letter {/ * 设置首字样式 * /
                   font - size: 24px; background - color: white;
                   border - radius: 5px;box - shadow: 0px 0px 10px 5px gray;}
   p { padding - left: 15px; padding - right: 10px; font - size: 14px;}
   body {background - color: lightgray;}
 </style></head>
<body><div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县
滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中
国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div>
<div><p>徐 海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会
夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红
军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background 属性用于一次性设置所有与背景相关的属性,如果其属性值是 repeating-radial-gradient,将呈现可重复任意次数的中心放射渐变背景。repeating-radial-gradient 的语法格式如下。

```
repeating - radial - gradient([ <position>,]? [ [ <shape> || <size> ] | <shape - size>{2},]? <color -
stop>[, <color - stop>] + )
```

在此实例中,repeating-radial-gradient(at bottom left, yellow, lightblue 2%, lightgreen 3%, lightcyan 15%)中的 at bottom left 表示放射中心是左下角,以中心为起点,按照所示比例渐变的颜色依次是 yellow、lightblue、lightgreen、lightcyan。

此实例的源文件名是 myHtmlB060.html。

223　为元素设置重复的线性渐变背景

此实例主要在 CSS 样式中设置元素的 background 属性为 repeating-linear-gradient,从而使元素(div)的背景呈现多次重复的线性渐变颜色。当在 Google Chrome 浏览器中显示该页面时,元素背景将按照 white、lightgreen、lightseagreen 的渐变颜色进行多次重复,效果如图 223-1 所示。有关此实例的主要代码如下。

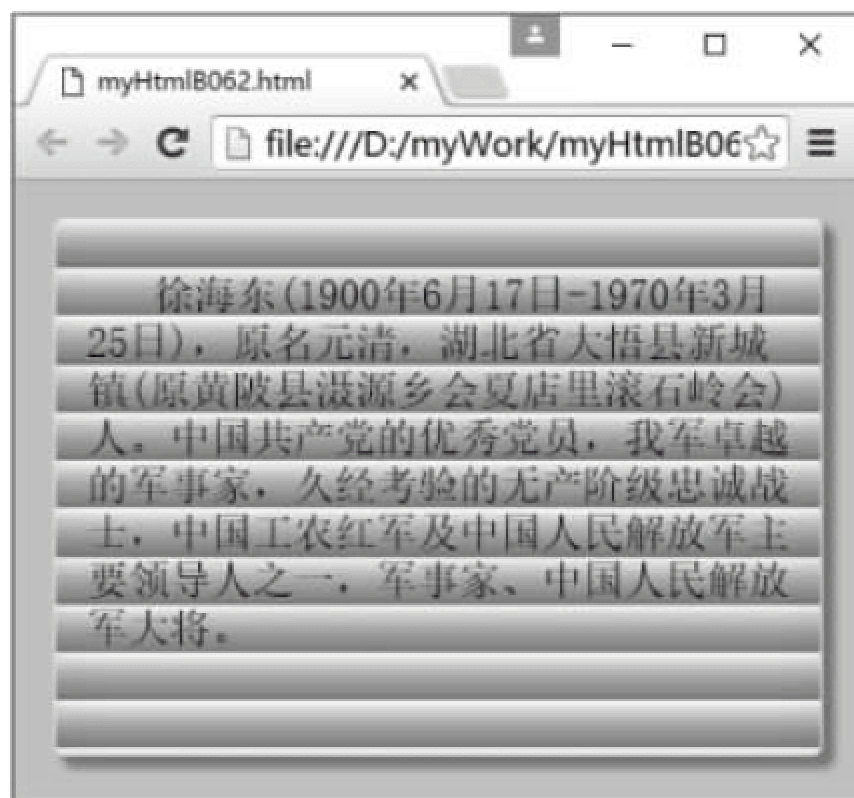


图　223-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block; width: 370px; height: 258px; margin: 10px;
       text-indent: 2em;background-color: #FFF;border: 1px solid #EEE;
       box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
       position: relative; border-radius: 5px;
       /* 设置重复的线性渐变背景 */
       background: repeating-linear-gradient(to bottom, white, lightgreen 5%,lightseagreen 9%);}
  p {padding-left: 15px;padding-right: 15px;padding-top: 5px;font-size: 20px;}
  body {background-color: lightgray;}
 </style></head>
<body><div><p>徐海东(1900年6月17日-1970年3月25日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background属性用于一次性设置所有与背景相关的属性,当该属性值为repeating-linear-gradient时将呈现重复的线性渐变背景颜色。repeating-linear-gradient的语法格式如下。

```
repeating-linear-gradient([<point>,]?<color-stop>[,<color-stop>]+)
```

各参数的说明如下。

(1) <point>: [left | right]? [top | bottom]? || <angle>?

其中,left用于设置左边为渐变起点的横坐标值;right用于设置右边为渐变起点的横坐标值;top用于设置顶部为渐变起点的纵坐标值;bottom用于设置底部为渐变起点的纵坐标值;<angle>用于以角度值指定渐变的方向(或角度)。

(2) <color-stop>: <color> [<length> | <percentage>]?

其中,<color>指定颜色;<length>表示用长度值指定起止色位置,不允许为负值;<percentage>表示用百分比指定起止色位置。

在此实例中,background: repeating-linear-gradient(to bottom, white, lightgreen 5%, lightseagreen 9%)中的to bottom表示渐变方向是从上向下,如果是to top,则正好相反,表示从下向上。

此实例的源文件名是myHtmlB062.html。

224 以角度方式设置重复的线性渐变背景

此实例主要在CSS样式中设置元素的background属性为repeating-linear-gradient,并设置旋转角度为45°,从而使元素(div)的背景在对角线上呈现多次重复的线性渐变颜色。当在Google Chrome浏览器中显示该页面时,元素背景将按照white、lightgreen、lightseagreen的渐变颜色在对角线上进行多次重复,效果如图224-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block;width: 370px; height: 235px;margin: 10px;
       text-indent: 2em;background-color: #FFF;border: 1px solid #EEE;
       box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
       position: relative;border-radius: 5px;
```

```
        /* 以角度方式设置重复的线性渐变背景 */
        background:repeating-linear-gradient(45deg,white, lightgreen 5%,lightseagreen 9%);}
   p {padding-left: 15px;padding-right: 15px;padding-top: 5px;font-size: 20px;}
   body {background-color: lightgray;}
  </style></head>
 <body>
  <div><p>徐海东(1900 年 6 月 17 日 - 1970 年 3 月 25 日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background 属性用于一次性设置所有与背景相关的属性,当该属性值为 repeating-linear-gradient 时将呈现重复的线性渐变背景颜色。线性渐变通常按照水平或垂直方向进行渐变,但是当起始位置是角度值时则可以按照一定的角度进行旋转,在此实例中,background: repeating-linear- gradient (45deg, white, lightgreen 5%, lightseagreen 9%)中的角度是 45°,因此在元素的对角线上呈现多次重复的线性渐变颜色。

此实例的源文件名是 myHtmlB063.html。

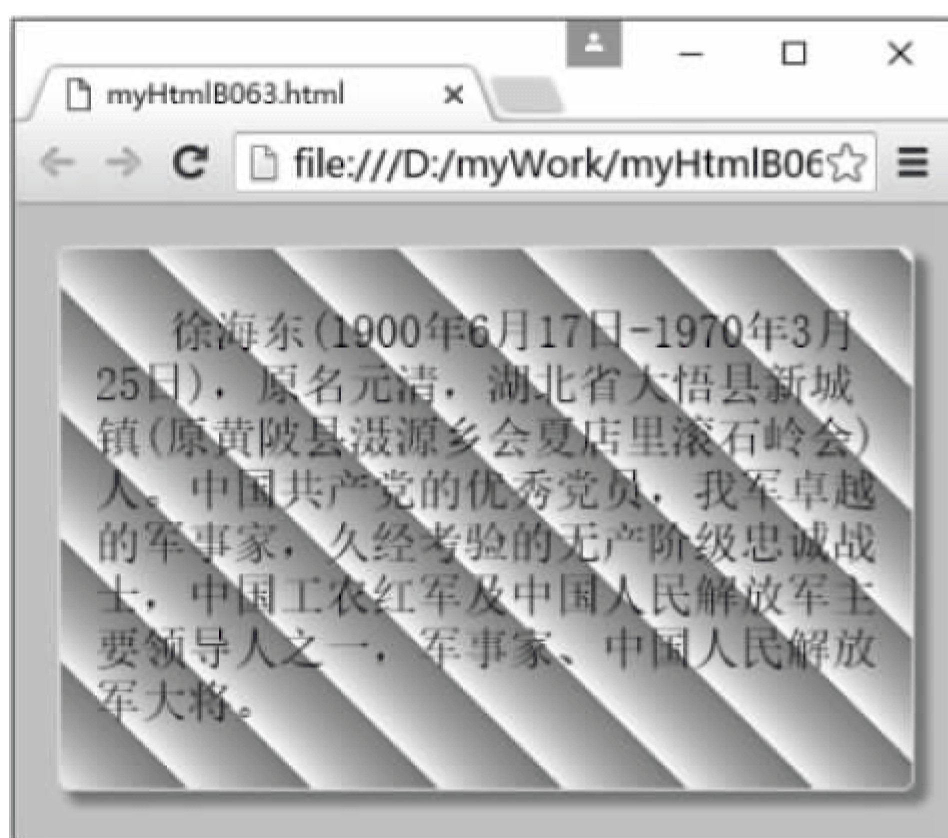


图　224-1

225　使用线性渐变方法创建波纹带状背景

此实例主要使用多个线性渐变方法创建背景图像,从而使背景呈现波纹带状的效果。当在 Google Chrome 浏览器中显示该页面时,黑白相间的波纹带状背景效果如图 225-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /* 创建波纹带状的背景 */
  .myWaveStripe { background: linear-gradient(135deg, black 25%, transparent 25%) -50px 0,
                  linear-gradient(225deg, black 25%, transparent 25%) -50px 0,
                  linear-gradient(315deg, black 25%, transparent 25%),
                  linear-gradient(45deg, black 25%, transparent 25%);
                  background-size: 40px 40px;}
```

```
    div { position: absolute; width: 270px; height: 240px;
        left:65px;top:15px;border - radius: 5px;background - image: url(img/B274.jpg);
          /*以椭圆形状裁剪背景图像*/
          - webkit - clip - path: ellipse(30 % 45 % at 50 % 50 %); }
  </style></head>
 <body class = "myWaveStripe"><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，linear-gradient()方法用于以线性渐变方式创建图像，为了实现波纹带状背景图像，此实例使用了4次该方法。此实例最奇妙的地方就是采用了背景图像的默认重复模式，即background-repeat:repeat，因为是默认值，所以没有在代码中出现。但是，如果仅允许背景图像在水平方向上重复，即background-repeat:repeat-x，则背景图像的效果如图225-2所示；如果仅允许背景图像在垂直方向上重复，即background-repeat:repeat-y，则背景图像的效果如图225-3所示；如果不允许背景图像重复，即background-size: 500px 500px; background-repeat: no-repeat，则背景图像的效果如图225-4所示，这就是4次使用linear-gradient()方法创建的图像。因此实现这种效果不仅与图像原本的形状有关，而且与重复模式和图像尺寸也有很大的关系。

此实例的源文件名是myHtmlB274.html。

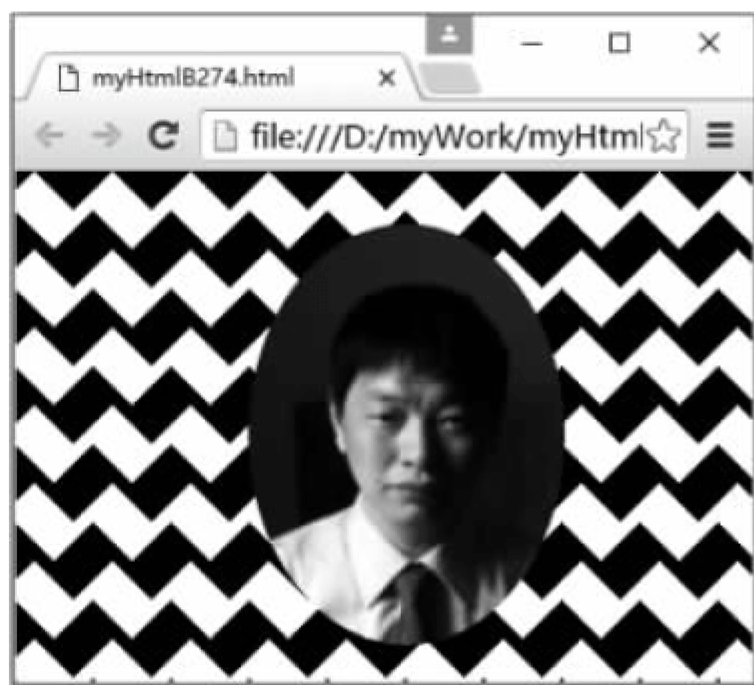


图 225-1



图 225-2



图 225-3



图 225-4

226 使用线性渐变方法创建箭头风格背景

此实例主要使用多个线性渐变方法创建背景图像，从而使背景呈现箭头风格的效果。当在 Google Chrome 浏览器中显示该页面时，红色的箭头沿着对角线方向从右上角向左下角重复，蓝色的箭头则沿着对角线方向从左下角向右上角重复，如图 226-1 所示。有关此实例的主要代码如下。



图 226-1

```
<! doctype html ><html><head><meta charset = "UTF - 8">
 <style>
  /* 创建箭头风格的背景 */
  .myArrowStripe { width: 400px; height:250px; border - radius:.5em;
                   background: linear - gradient(45deg, red 45px, transparent 45px)64px 64px,
                   linear - gradient(45deg, red 45px, transparent 45px,transparent 91px, blue 91px, blue
                   135px, transparent 135px),
                   linear - gradient( - 45deg, red 23px, transparent 23px, transparent 68px,red 68px,red
                   113px,transparent 113px,transparent 158px,red 158px);
                   background - color:blue; background - size: 128px 128px;}
  .myBox { position: absolute; width: 270px; height: 240px; border - radius: 5px;
          left: 135px; top: 15px; background - image: url(img/B277.jpg);
          /* 以椭圆形状裁剪背景图像 */
          - webkit - clip - path: ellipse(30 % 45 % at 30 % 50 % ); }
 </style></head>
<body><div class = "myArrowStripe" ><div class = "myBox" ></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，linear-gradient()方法用于以线性渐变方式创建图像，为了实现箭头风格的背景图像，此实例使用了 3 次该方法。此实例最奇妙的地方就是采用了背景图像的默认重复模式，即 background-repeat：repeat，因为是默认值，所以没有在代码中出现。如果设置其他重复模式，例如 background-repeat：repeat-y、background-repeat：repeat-x、background-repeat：no-repeat 等，则不会产生此种箭头效果。

此实例的源文件名是 myHtmlB277.html。

227 以滤色模式显示渐变背景和图像背景

此实例主要设置 background-blend-mode 属性为 screen，并在 background 属性中同时设置渐变颜色背景和图像背景，从而实现图像背景和渐变颜色背景以滤色模式显示。当在 Google Chrome 浏览器中显示该页面时，单击“渐变背景”按钮，则下面的背景将以由绿色到白色的渐变色显示，如图 227-1 所示；单击“图像背景”按钮，则下面的背景将显示一幅图像，如图 227-2 所示；单击“滤色背景”按钮，则将渐变色背景和图像背景以滤色模式处理后显示，如图 227-3 所示。有关此实例的主要代码如下。



图 227-1



图 227-2



图 227-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnGradient").click(function() {//渐变背景
```

```
        $(".myDiv").css("background", "linear-gradient(limegreen, transparent)");
      });
      $("#myBtnImage").click(function() {//图像背景
       $(".myDiv").css("background", "url(img/B152B.jpg)");
      });
      $("#myBtnScreen").click(function() {//滤色背景
       $(".myDiv").css("background", "linear-gradient(limegreen, transparent),url(img/B152B.jpg)");
      });});
    </script>
    <style type="text/css">
     .myDiv { width: 400px; height:250px;line-height: 250px; font-size: 38px;
             text-align: center; border-radius: 10px;
             background:linear-gradient(limegreen, transparent),url(img/B152B.jpg);
             background-blend-mode: screen; background-size: 100% 100%; }
     button{ width:124px; margin: 2px;}
    </style></head>
   <body><div>
        <button id="myBtnGradient">渐变背景</button>
        <button id="myBtnImage">图像背景</button>
        <button id="myBtnScreen">滤色背景</button>
   </div>
   <div class="myDiv">炫酷实例集锦</div>
   </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。background-blend-mode：screen 表示背景层的图像和渐变色以 screen(滤色)模式混合显示，在采用滤色模式混合图像时将查看每个通道的颜色信息，并将混合色的互补色复合，结果总是显示较亮的颜色，即任何颜色和白色进行滤色都得到白色，任何颜色和黑色进行滤色都会保留原来的颜色，在此实例中经过滤色后显示最多的就是绿色。需要注意的是，当使用 background-blend-mode:screen 时，background 属性中必须有两个或两个以上的背景图像。background-blend-mode 属性支持的混合模式有 normal、multiply、screen、overlay、darken、lighten、color-dodge、color-burn、hard-light、soft-light、difference、exclusion、hue、saturation、color、luminosity，其中每一种混合模式都有其各自的计算模式。

此实例的源文件名是 myHtmlB279.html。

228 通过重复线性渐变实现信纸风格背景

此实例主要设置 background 属性为 repeating-linear-gradient，从而使背景呈现细条纹信纸风格的特殊效果。当在 Google Chrome 浏览器中显示该页面时，红、白、绿、白相间的细条纹信纸风格的背景效果如图 228-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myStripe { margin: 10px auto;width: 400px; height: 250px;border-radius: 10px;
             /*通过重复的多级渐变实现细条纹信纸风格的背景*/
             background: repeating-linear-gradient(45deg, red 0px, red 1px,white 2px, white 5px,
             green 6px, green 7px, white 4px, white 19px);}
```

```
   .myBox { margin: 10px auto; width: 400px;height: 250px;
          background-image: url(img/B003.jpg); background-size: 100% 100%;
          /* 以椭圆形状裁剪背景图像 */
          -webkit-clip-path: ellipse(30% 35% at 50% 50%);}
  </style></head>
 <body><div class="myStripe"><div class="myBox"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background: repeating-linear-gradient(45deg, red 0px, red 1px, white 2px, white 5px, green 6px, green 7px, white 4px, white 19px)用于实现红、白、绿、白重复的背景，各种颜色后面的像素值主要用于控制渐变范围，45deg 则表示所有颜色的渐变方向均为 45°，即对角线方向。

此实例的源文件名是 myHtmlB280.html。

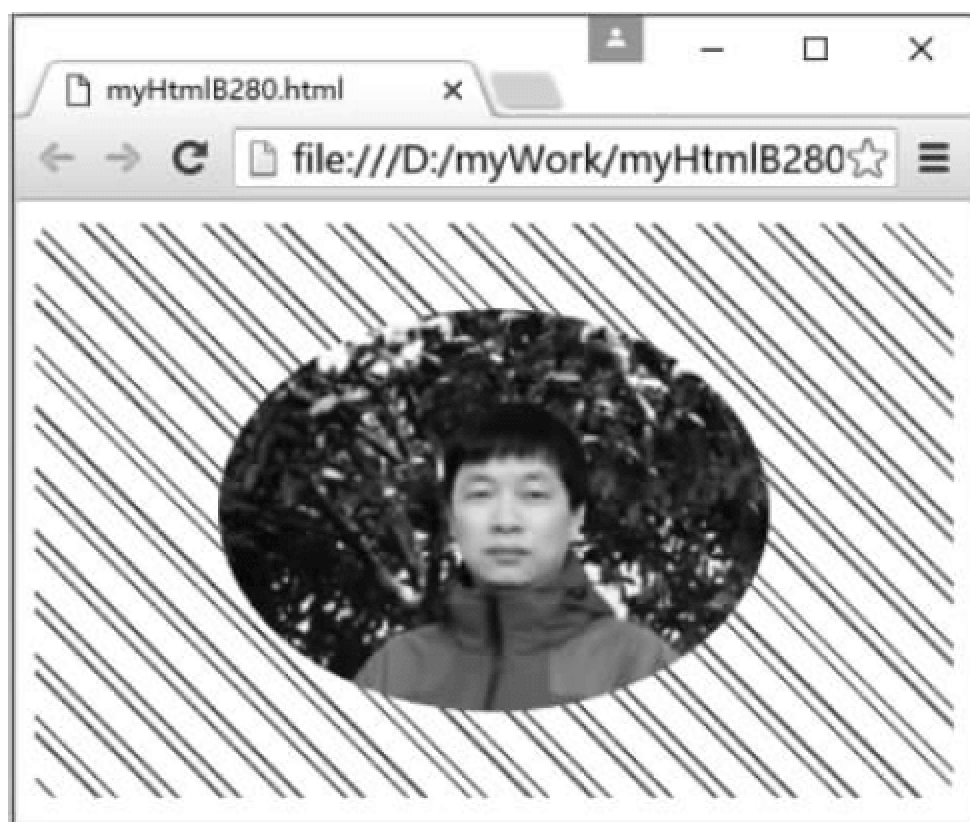


图 228-1

229 使用多个径向渐变实现太极图案背景

此实例主要通过使用多个 radial-gradient()创建不同的圆形来设置背景，从而使背景呈现太极图案的效果。当在 Google Chrome 浏览器中显示该页面时，创建的太极图案背景如图 229-1 所示。有关此实例的主要代码如下。

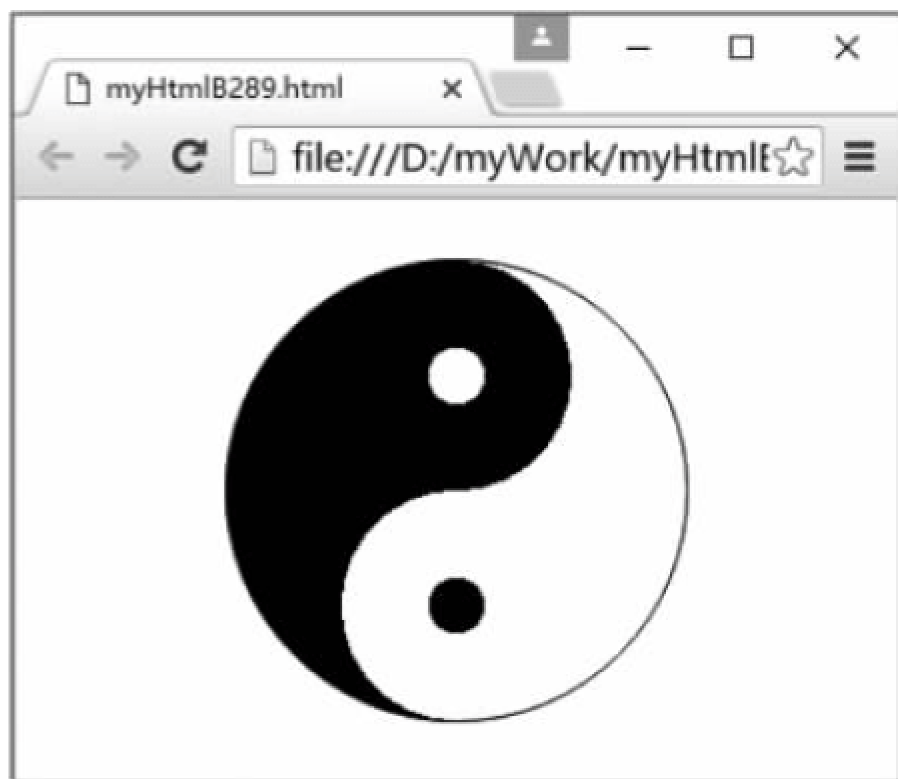


图 229-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myTaichi { position: absolute; top: 50%;left: 50%;
              width: 200px; height: 200px;
              transform: translate(-50%, -50%); border-radius: 50%;
             /*绘制太极图案的空心外圆*/
              border:solid 1px black;
              background-image:
                /*绘制太极图案下面的小黑点*/
                radial-gradient(black 12.5px, transparent 12.5px),
                /*绘制太极图案上面的小白点*/
                radial-gradient(white 12.5px, transparent 12.5px),
                /*绘制太极图案下面的小黑点的外层白圆*/
                radial-gradient(white 50px, transparent 50px),
                /*绘制太极图案上面的小白点的外层黑圆*/
                radial-gradient(black 50px, transparent 50px),
                /*将最外层的外圆的左半部分用黑色填充,右半部分用白色填充*/
                linear-gradient(90deg, black 100px, white 100px);
                /*4个坐标分别对应前面的4个圆的中心点*/
   background-position: center 50px, center -50px, center 50px, center -50px;}
 </style></head>
<body><div class="myTaichi"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-image：radial-gradient(black 12.5px，transparent 12.5px)和 background-position：center 50px 用于在大圆中创建一个小黑点，并显示在大圆的下半部分，如图 229-2 所示，其他 3 个圆的实现原理与之类似；background-image：linear-gradient(90deg，black 100px，white 100px)和 background-position：0 通过使用两级线性渐变使大圆的左半部分显示黑色，右半部分显示白色，如图 229-3 所示。在 CSS3 中，radial-gradient()方法用于以径向渐变方式创建图像，linear-gradient()方法用于以线性渐变方式创建图像。

此实例的源文件名是 myHtmlB289.html。

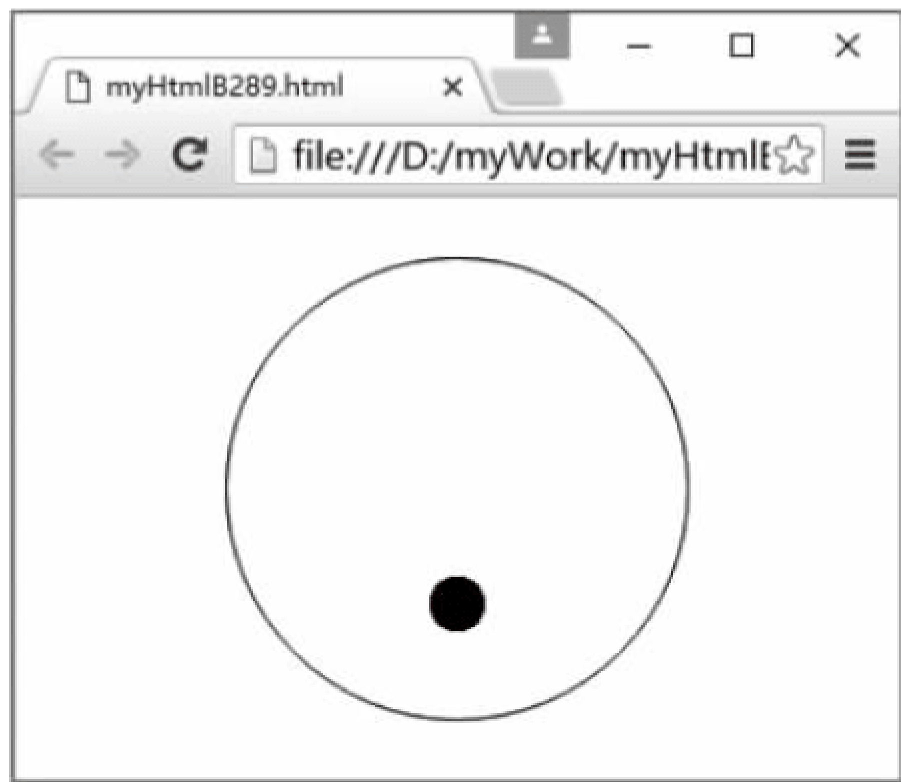


图　229-2

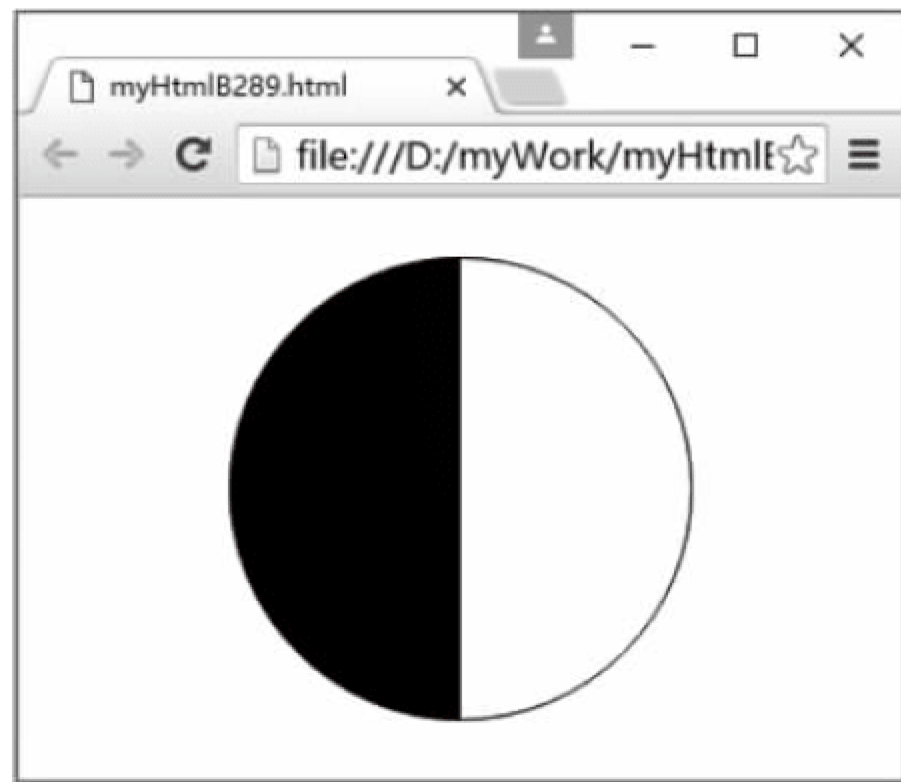


图　229-3

230　创建渐变背景实现带线条风格的稿纸

此实例主要设置 background 和-webkit-background-size 属性，从而以创建渐变背景的形式实现带线条风格的书信稿纸。当在 Google Chrome 浏览器中显示该页面时，带线条的书信稿纸效果如

图 230-1 所示。有关此实例的主要代码如下。

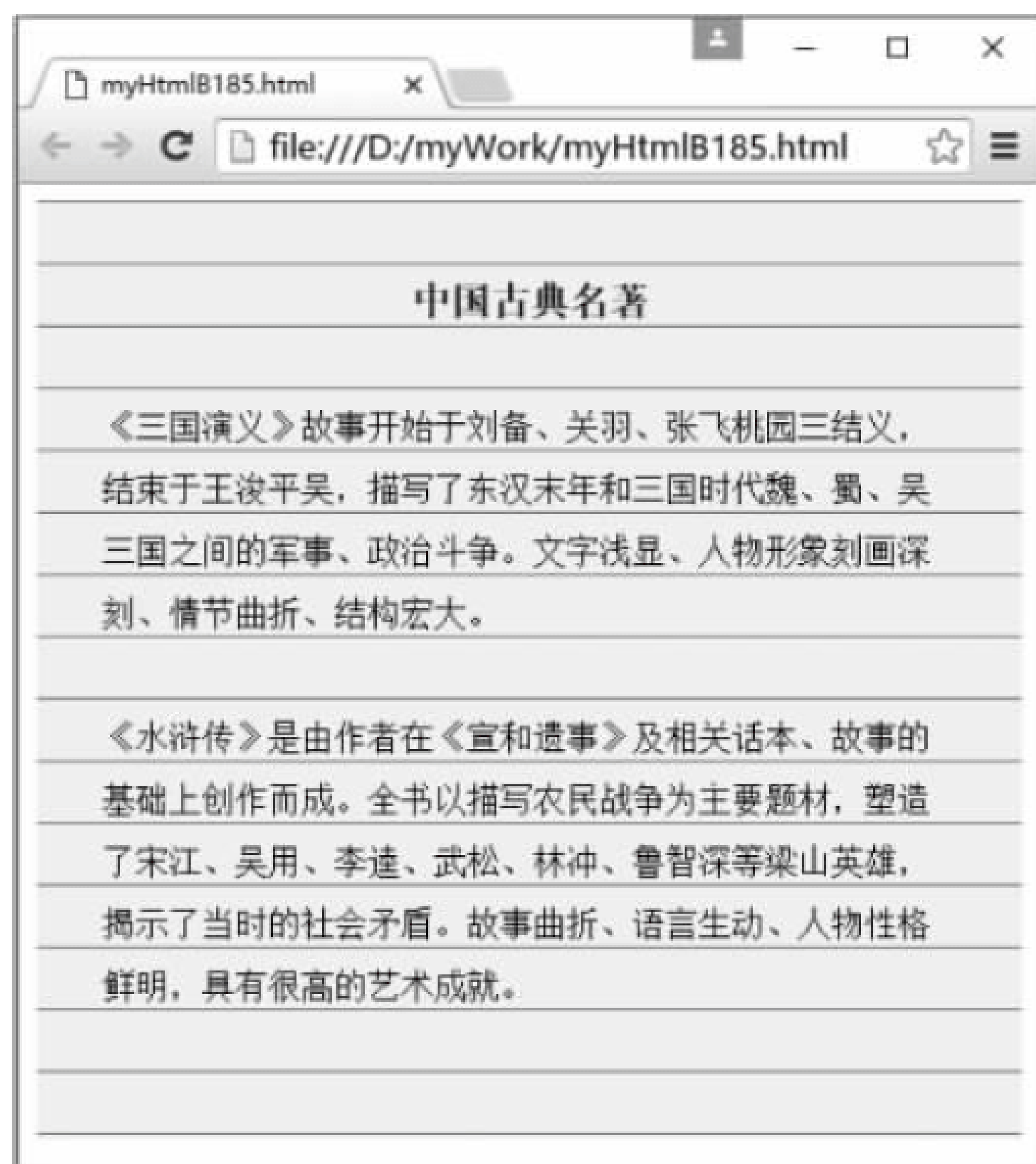


图 230-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .page {line-height: 30px; margin: 0 auto;padding: 15px 0 29px;
        background: -webkit-gradient( linear,left top, left bottom, from(green), color-stop(5%,
        lightgoldenrodyellow) ); -webkit-background-size: 100% 30px; }
  .page p { margin: 30px 0;padding: 0 2em;}
 </style></head>
<body><div class="page"><center><h3>中国古典名著</h3></center>
  <p>«三国演义»故事开始于刘备、关羽、张飞桃园三结义，结束于王浚平吴，描写了东汉末年和三国时代魏、蜀、吴三国之间的军事、政治斗争。文字浅显、人物形象刻画深刻、情节曲折、结构宏大。</p>
  <p>《水浒传》是由作者在《宣和遗事》及相关话本、故事的基础上创作而成的。全书以描写农民战争为主要题材，塑造了宋江、吴用、李逵、武松、林冲、鲁智深等梁山英雄，揭示了当时的社会矛盾。故事曲折、语言生动、人物性格鲜明，具有很高的艺术成就。</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-gradient(linear，left top，left bottom，from(green)，color-stop(5%，lightgoldenrodyellow))用于创建每行的渐变背景效果，读者试着将 5%修改为 75%即可明白这种特殊的渐变效果；-webkit-background-size：100% 30px 用于规定前面所创建的渐变背景在水平方向铺满，在垂直方向每隔 30px 重复一次，也可以直接写-webkit-background-size：30px，效果相同。在 CSS 中，background-size 属性用于规定背景图像的尺寸，该属性的语法格式如下。

```
background-size: length | percentage | cover | contain;
```

其中，各属性值的意义如下。

(1) length：表示设置背景图像的高度和宽度，第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto。

(2) percentage：表示以父元素的百分比来设置背景图像的宽度和高度，第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto。

(3) cover：表示把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。背景图像的某些部分也许无法显示在背景定位区域中。

(4) contain：表示把图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域。

此实例的源文件名是 myHtmlB185.html。

231　使用线性渐变实现背景隔行错色显示

此实例主要使用 linear-gradient()方法绘制两种颜色平分高度的背景图形，并设置 line-height 属性值为背景图形高度的一半，从而实现文本的背景颜色隔行交错显示的效果。当在 Google Chrome 浏览器中显示该页面时，文本的背景颜色以青色和粉色隔行交错显示的效果如图 231-1 所示。有关此实例的主要代码如下。

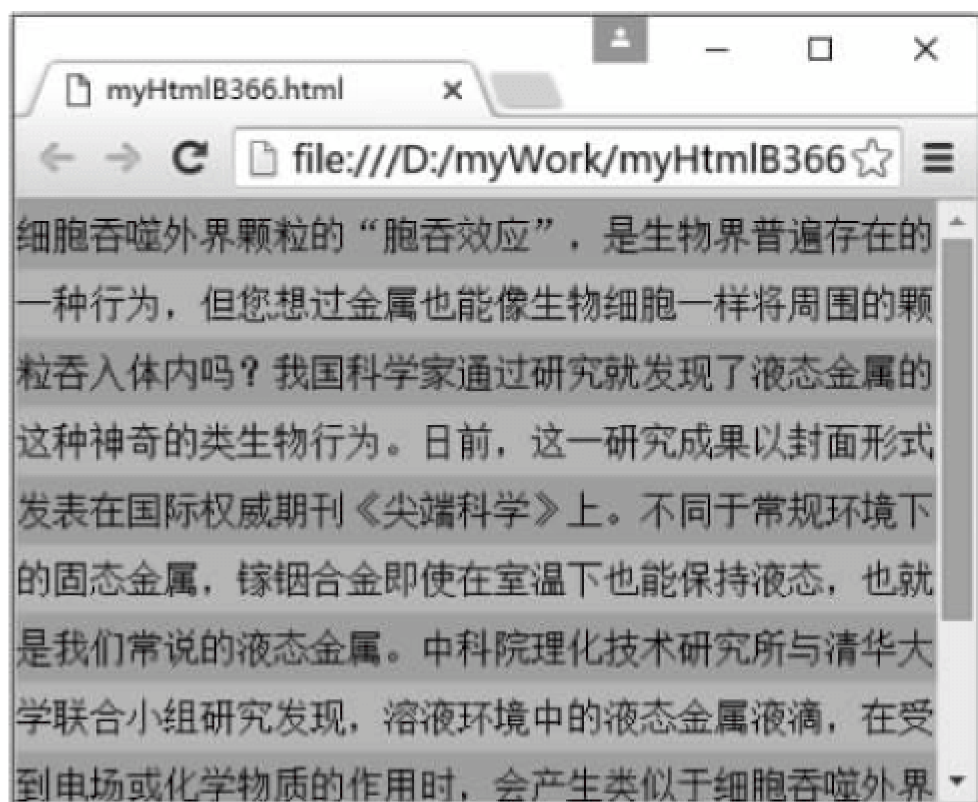


图　231-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   *{ margin: 0; padding: 0; }
  .myBox{ font-size: 16px;line-height: 30px;background-origin: content-box;
          /* 创建线性渐变背景图形 */
          background: linear-gradient(cyan 50%,pink 50%);
          /* 设置线性渐变背景图形的大小,高度为 line-height 的两倍 */
          background-size: 100% 60px; }
 </style></head>
<body><div class="myBox">
  <p>细胞吞噬外界颗粒的"胞吞效应",是生物界普遍存在的一种行为,但您想过金属也能像生物细胞一样将周围的颗粒吞入体内吗?我国科学家通过研究就发现了液态金属的这种神奇的类生物行为。日前,这一研究成果以封面形式发表在国际权威期刊«尖端科学»上。不同于常规环境下的固态金属,镓铟合金即使在室温下也能保持液态,也就是我们常说的液态金属。中科院理化技术研究所与清华大学联合小组研究发现,溶液环境中的液态金属液滴,在受到电场或化学物质的作用时,会产生类似于细胞吞噬外界颗粒的胞吞效应,能高效地将周围的颗粒吞入体内。这一发现也开辟了一条构筑高性能纳米金属流体材料的新途径。</p></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，line-height：30px 用于设置行间的距离（行高）为 30px；background：linear-gradient(cyan 50%，pink 50%)用于创建青色和粉色各占 50%

的背景图形；background-size：100% 60px 用于设置背景图形的高度为 60px，即 line-height 属性值的两倍，宽度等于盒子的宽度。

此实例的源文件名是 myHtmlB366.html。

232 在元素的对角线上设置线性渐变的背景色

此实例主要在 CSS 样式中设置元素的 background 属性为 linear-gradient，并设置其渐变方向为 to bottom left，从而使元素（div）的背景呈现从右上角向左下角线性渐变的颜色。当在 Google Chrome 浏览器中显示该页面时，这种在对角线上线性渐变的背景效果如图 232-1 所示。有关此实例的主要代码如下。

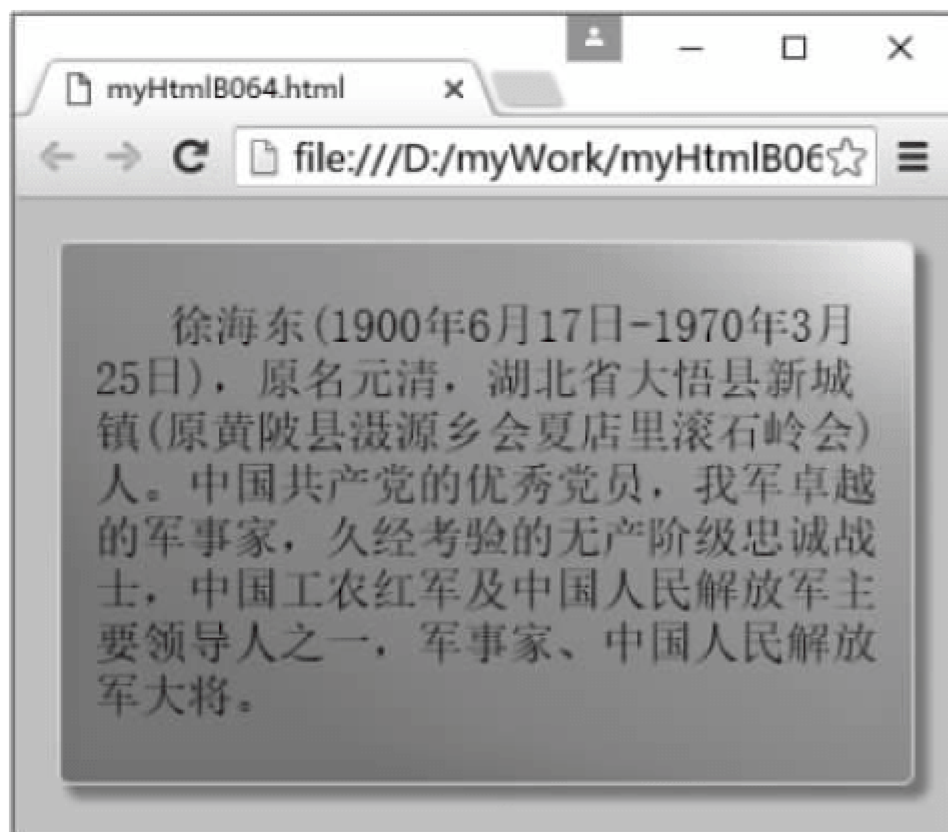


图 232-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block;width: 370px; height: 235px;margin: 10px;
      text-indent: 2em; background-color: #FFF; border: 1px solid #EEE;
      box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
      position: relative;border-radius: 5px;
      /* 设置从右上角到左下角的线性渐变背景 */
      background:linear-gradient(to bottom left,white, lightgreen 20%, lightseagreen 80%); }
  p {padding-left: 15px;padding-right: 15px;padding-top: 5px;font-size: 20px;}
  body {background-color: lightgray;}
 </style></head>
<body><div><p>徐海东(1900年6月17日-1970年3月25日),原名元清,湖北省大悟县新城镇(原黄陂县滠源乡会夏店里滚石岭会)人。中国共产党的优秀党员,我军卓越的军事家,久经考验的无产阶级忠诚战士,中国工农红军及中国人民解放军主要领导人之一,军事家、中国人民解放军大将。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background 属性用于一次性设置所有与背景相关的属性，如果此属性值为 linear-gradient，则将使背景呈现线性渐变效果。linear-gradient 的语法格式如下。

```
linear-gradient(direction, color-stop1, color-stop2, …);
```

其中，direction 默认为 to bottom，即从上向下的渐变。如果 direction 的值为 to bottom left，则渐变方向是从右上角向左下角，沿着对角线方向线性渐变；如果采用角度表示，to bottom left 相当于一

135deg。如果 direction 的值为 to bottom right，则渐变方向是从左上角向右下角，沿着对角线方向线性渐变；如果采用角度表示，to bottom right 相当于 135deg。

如果不想将渐变色的起点或终点设置为元素的顶端、底端、左边、右边、左上角、左下角、右上角或右下角，可以在起点色或终点色后边指定离渐变色起点或渐变色终点的偏离位置(不指定时默认值分别是 0% 和 100%)，例如 background: linear-gradient (to bottom left, white, lightgreen 20%, lightseagreen 80%)。

此实例的源文件名是 myHtmlB064.html。

233　透明显示在元素背景中叠加的两幅图像

此实例主要通过改变-webkit-cross-fade(url(img/B156A.jpg), url(img/B156B.jpg), 50%)方法中的百分比值实现以不同的透明度显示在元素背景中叠加的两幅图像。当在 Google Chrome 浏览器中显示该页面时，使用鼠标拖动滑块修改透明度值(即百分比值)，即能够以不同的透明度显示两幅图像的叠加效果，如图 233-1 所示。有关此实例的主要代码如下。



图　233-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   function myFunc() {//获取滑块的当前值并转换成 float
    var myFade=parseFloat($("#myFade").prop("value"));
    //根据滑块值重置图像的透明度
     $("div").css("background-image","-webkit-cross-fade(url(img/B156A.jpg), url(img/B156B.
jpg), "+myFade); }
   setInterval(myFunc,1);
});
 </script>
 <style type="text/css">
  div{ width:450px; height:300px; border-radius: 10px;background-image: -webkit-cross-fade(url
(img/B156A.jpg), url(img/B156B.jpg), 50%); }
```

```
    input{ width: 230px;}
  </style></head>
<body><center>拖动滑块改变图像的透明度: <input type="range" id="myFade" min="0" max="1" value=
"0.5" step="0.05" /><br><br><div></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-image: -webkit-cross-fade(url(img/B156A.jpg), url(img/B156B.jpg), 50%)表示以 50%的透明度控制第二幅图像与第一幅图像叠加显示,目前的测试表明,透明度百分比值只作用于第二幅图像上。

此实例的源文件名是 myHtmlB177.html。

234 以叠加和滤色模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 overlay 和 screen,从而实现以叠加和滤色模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时,两幅图像在未混合前只显示一幅图像,单击"叠加模式"按钮,则两幅图像经过叠加模式混合后的效果如图 234-1 所示;单击"滤色模式"按钮,则两幅图像经过滤色模式混合后的效果如图 234-2 所示。有关此实例的主要代码如下。



图 234-1



图 234-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnOverlay").click(function() {//叠加模式
     $(".box").css("background-blend-mode", "overlay"); });
    $("#myBtnScreen").click(function() {//滤色模式
     $(".box").css("background-blend-mode", "screen"); });});
 </script>
 <style type="text/css">
   .box {width: 450px;height: 300px;border-radius: 10px;margin: 3px;background: url(img/B151A.jpg)
no-repeat center, url(img/B151B.jpg) no-repeat center;}
   input { width: 220px; border-radius: 2px; padding: 3px; margin: 2px; }
 </style></head>
```

```
<body><div><input type="button" value="叠加模式" id="myBtnOverlay"/>
     <input type="button" value="滤色模式" id="myBtnScreen"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode","overlay")表示元素背景层的两幅图像以叠加模式混合显示，在采用叠加模式混合图像时，复合或过滤颜色取决于基色(backdrop)的值，图像或颜色在现有的像素上叠加，同时保留基色的明暗度。$(".box").css("background-blend-mode","screen")表示元素背景层的两幅图像以滤色模式混合显示，在采用滤色模式混合图像时将查看每个通道的颜色信息，并将混合色的互补色和基色复合，结果总是较亮的颜色，任何颜色和白色进行滤色都得到白色，任何颜色和黑色进行滤色都会保留原来的颜色。

此实例的源文件名是 myHtmlB151.html。

235 以差值和排除模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 difference 和 exclusion，从而实现以差值和排除模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击“排除模式”按钮，则两幅图像经过排除模式混合后的效果如图 235-1 所示；单击“差值模式”按钮，则两幅图像经过差值模式混合后的效果如图 235-2 所示。有关此实例的主要代码如下。

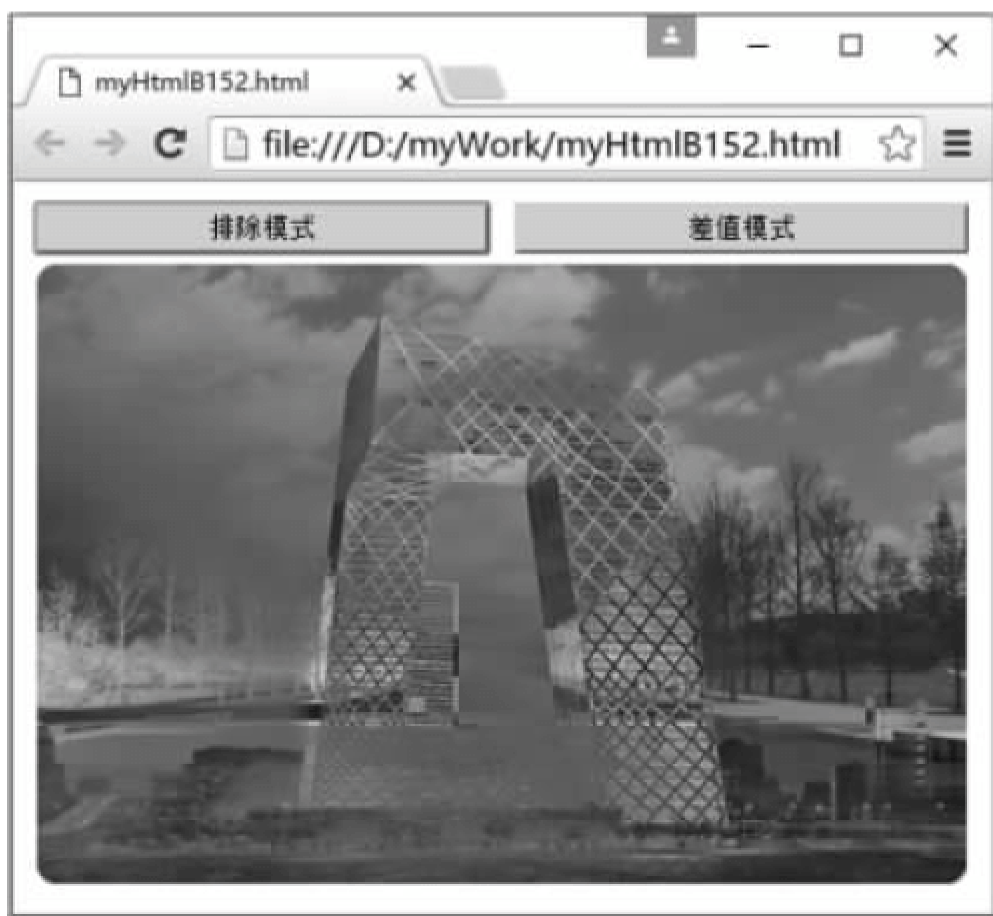


图 235-1

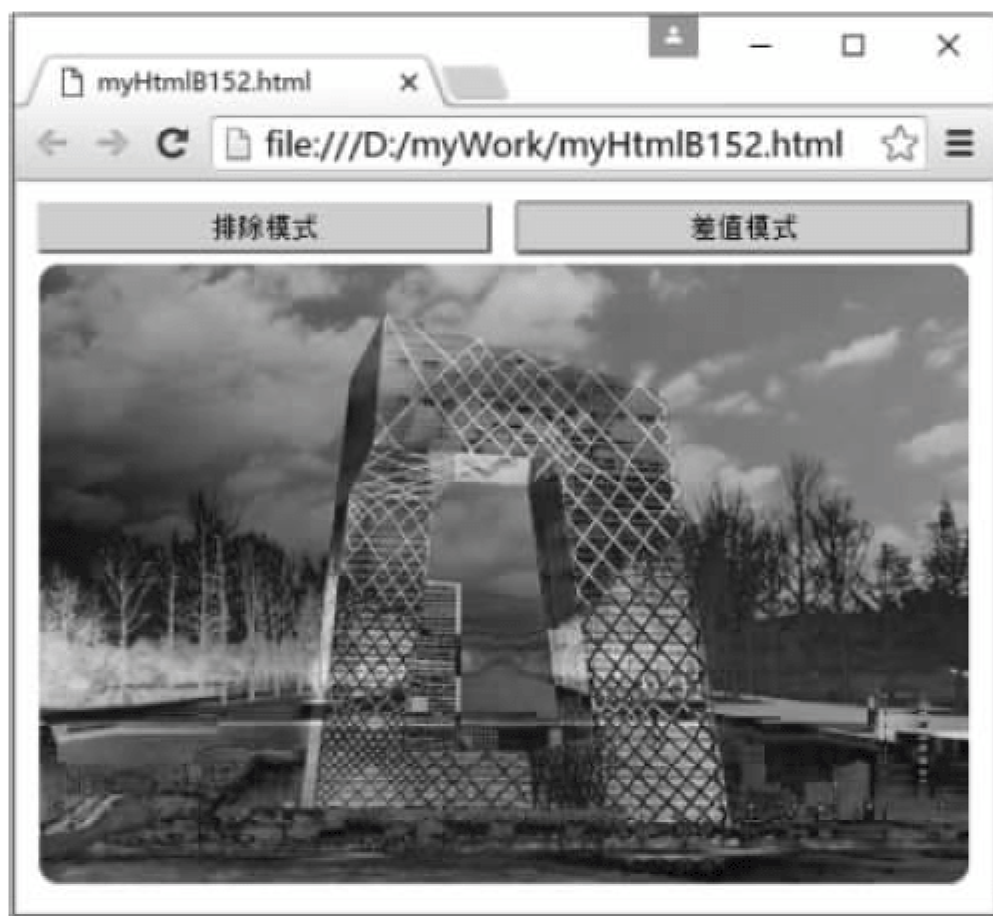


图 235-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnExclusion").click(function() {//排除模式
    $(".box").css("background-blend-mode", "exclusion"); });
   $("#myBtnDifference").click(function() {//差值模式
    $(".box").css("background-blend-mode", "difference"); });});
 </script>
 <style type="text/css">
```

```
    .box { width: 450px; height: 300px;border-radius: 10px;margin: 3px;
          background: url(img/B152A.jpg) no-repeat center, url(img/B152B.jpg) no-repeat center;}
    input { width: 220px;border-radius: 2px; padding: 3px; margin: 2px;}
  </style></head>
<body><div><input type="button" value="排除模式" id="myBtnExclusion"/>
     <input type="button" value="差值模式" id="myBtnDifference"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode", "difference")表示元素背景层的两幅图像以差值模式混合显示，差值模式在混合图像时将查看每个通道的颜色信息，并从基色中减去混合色或从混合色中减去基色，具体取决于哪一个颜色的亮度值更大，与白色混合将反转基色值，与黑色混合则不产生变化。$(".box").css("background-blend-mode", "exclusion")表示元素背景层的两幅图像以排除模式混合显示，排除模式在混合图像时将插入一种与差值模式相似但对比度更低的效果，与白色混合将反转基色值，与黑色混合不发生变化。

此实例的源文件名是 myHtmlB152.html。

236 以强光和柔光模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 hard-light 和 soft-light，从而实现以强光和柔光模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击“强光模式”按钮，则两幅图像经过强光模式混合后的效果如图 236-1 所示；单击“柔光模式”按钮，则两幅图像经过柔光模式混合后的效果如图 236-2 所示。有关此实例的主要代码如下。



图 236-1



图 236-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
     $("#myBtnHard").click(function() {//强光模式
       $(".box").css("background-blend-mode", "hard-light"); });
```

```
    $ ("#myBtnSoft").click(function() {//柔光模式
      $ (".box").css("background-blend-mode", "soft-light"); });});
  </script>
  <style type="text/css">
   .box { width: 450px; height: 300px;border-radius: 10px; margin: 3px;background: url(img/B153A.jpg)
 no-repeat center, url(img/B153B.jpg) no-repeat center; }
   input { width: 220px; border-radius: 2px; padding: 3px; margin: 2px;}
  </style></head>
 <body><div><input type="button" value="强光模式" id="myBtnHard"/>
      <input type="button" value="柔光模式" id="myBtnSoft"/></div>
 <div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode","hard-light")表示元素背景层的两幅图像以强光模式混合显示，强光模式在混合图像时将过滤颜色，具体取决于混合色，此效果与耀眼的聚光灯照在图像上相似，如果混合色比 50%的灰度色亮，则图像变亮；如果混合色比 50%的灰度色暗，则图像变暗。$(".box").css("background-blend-mode","soft-light")表示元素背景层的两幅图像以柔光模式混合显示，柔光模式在混合图像时将使颜色变亮或变暗，具体取决于混合色，此效果与发散的聚光灯照在图像上的效果类似，如果混合色比 50%的灰度色亮，则图像变亮，就像被减淡一样；如果混合色比 50%的灰度色暗，则图像变暗，就像被加深了一样。

此实例的源文件名是 myHtmlB153.html。

237 以加深和减淡模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 color-dodge 和 color-burn，从而实现以颜色减淡和颜色加深模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击"颜色减淡模式"按钮，则两幅图像经过颜色减淡模式混合后的效果如图 237-1 所示；单击"颜色加深模式"按钮，则两幅图像经过颜色加深模式混合后的效果如图 237-2 所示。有关此实例的主要代码如下。

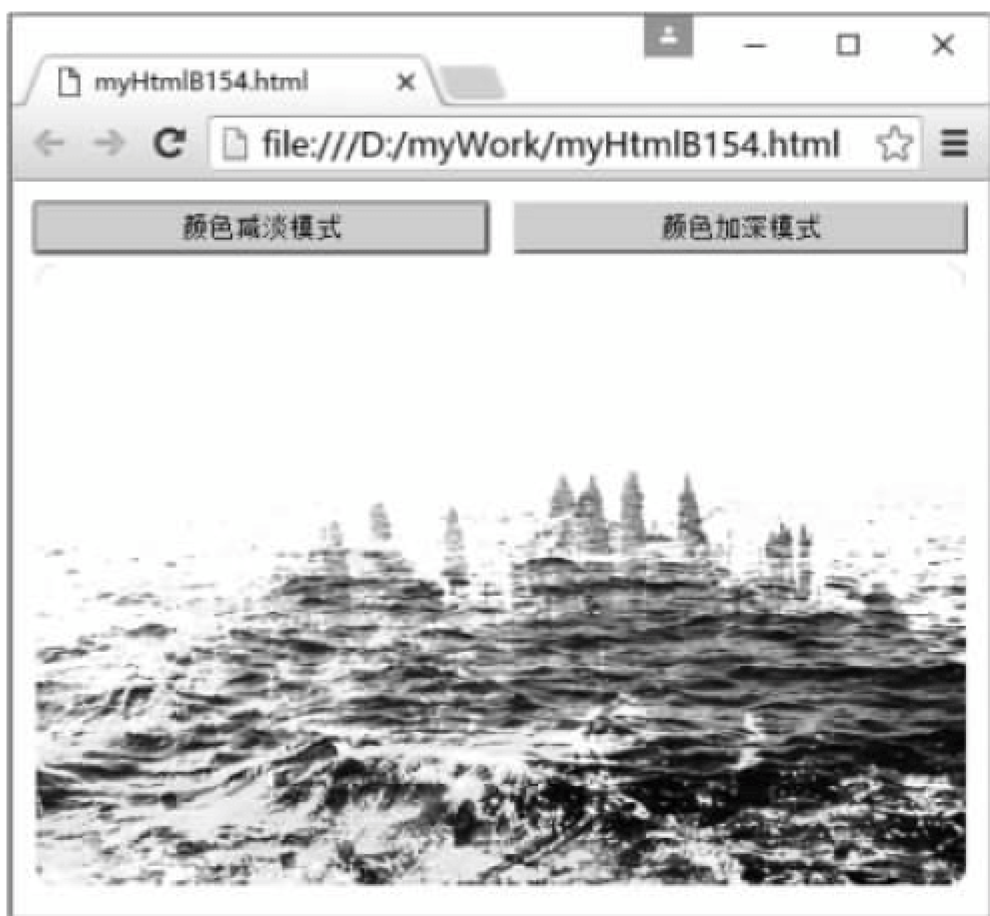


图 237-1



图 237-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnDodge").click(function() {//颜色减淡模式
    $(".box").css("background-blend-mode", "color-dodge"); });
   $("#myBtnBurn").click(function() {//颜色加深模式
    $(".box").css("background-blend-mode", "color-burn"); });});
 </script>
 <style type="text/css">
  .box { width: 450px; height: 300px;border-radius: 10px; margin: 3px;background: url(img/B154B.jpg)
no-repeat center, url(img/B154A.jpg) no-repeat center; }
  input {width: 220px; border-radius: 2px; padding: 3px; margin: 2px; }
 </style></head>
<body><div><input type="button" value="颜色减淡模式" id="myBtnDodge"/>
     <input type="button" value="颜色加深模式" id="myBtnBurn"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode","color-dodge")表示元素背景层的两幅图像以颜色减淡模式混合显示，颜色减淡模式在混合图像时将查看每个通道中的颜色信息，并通过减小对比度使基色变亮以反映混合色，与黑色混合不发生任何变化。$(".box").css("background-blend-mode","color- burn")表示元素背景层的两幅图像以颜色加深模式混合显示，颜色加深模式在混合图像时将查看每个通道中的颜色信息，并通过增加对比度使基色变暗以反映混合色，与白色混合不会发生变化。

此实例的源文件名是 myHtmlB154.html。

238 以增暗和增亮模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 darken 和 lighten，从而实现以增暗和增亮模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击“增暗模式”按钮，则两幅图像经过增暗模式混合后的效果如图 238-1 所示；单击“增亮模式”按钮，则两幅图像经过增亮模式混合后的效果如图 238-2 所示。有关此实例的主要代码如下。



图 238-1



图 238-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnDarken").click(function() {//增暗模式
    $(".box").css("background-blend-mode", "darken"); });
   $("#myBtnLighten").click(function() {//增亮模式
    $(".box").css("background-blend-mode", "lighten"); });});
 </script>
 <style type="text/css">
  .box {width: 450px;height: 300px; border-radius: 10px;margin: 3px;background: url(img/B155A.jpg)
no-repeat center, url(img/B155B.jpg) no-repeat center;}
  input {width: 220px;border-radius: 2px;padding: 3px; margin: 2px;}
 </style></head>
<body><div><input type="button" value="增暗模式" id="myBtnDarken"/>
     <input type="button" value="增亮模式" id="myBtnLighten"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode", "darken")表示元素背景层的两幅图像以增暗模式混合显示，增暗模式在混合图像时将查看每个通道的颜色信息，并选择基色或混合色中较暗的颜色作为结果色，比混合色亮的像素被替换，比混合色暗的像素保持不变。$(".box").css("background-blend-mode", "lighten")表示元素背景层的两幅图像以增亮模式混合显示，增亮模式在混合图像时将选择基色或混合色中较亮的颜色作为结果色，比混合色暗的像素被替换，比混合色亮的像素保持不变。

此实例的源文件名是 myHtmlB155.html。

239 以色相和亮度模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 hue 和 luminosity，从而实现以色相和亮度模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击“色相模式”按钮，则两幅图像经过色相模式混合后的效果如图 239-1 所示；单击“亮度模式”按钮，则两幅图像经过亮度模式混合后的效果如图 239-2 所示。有关此实例的主要代码如下。

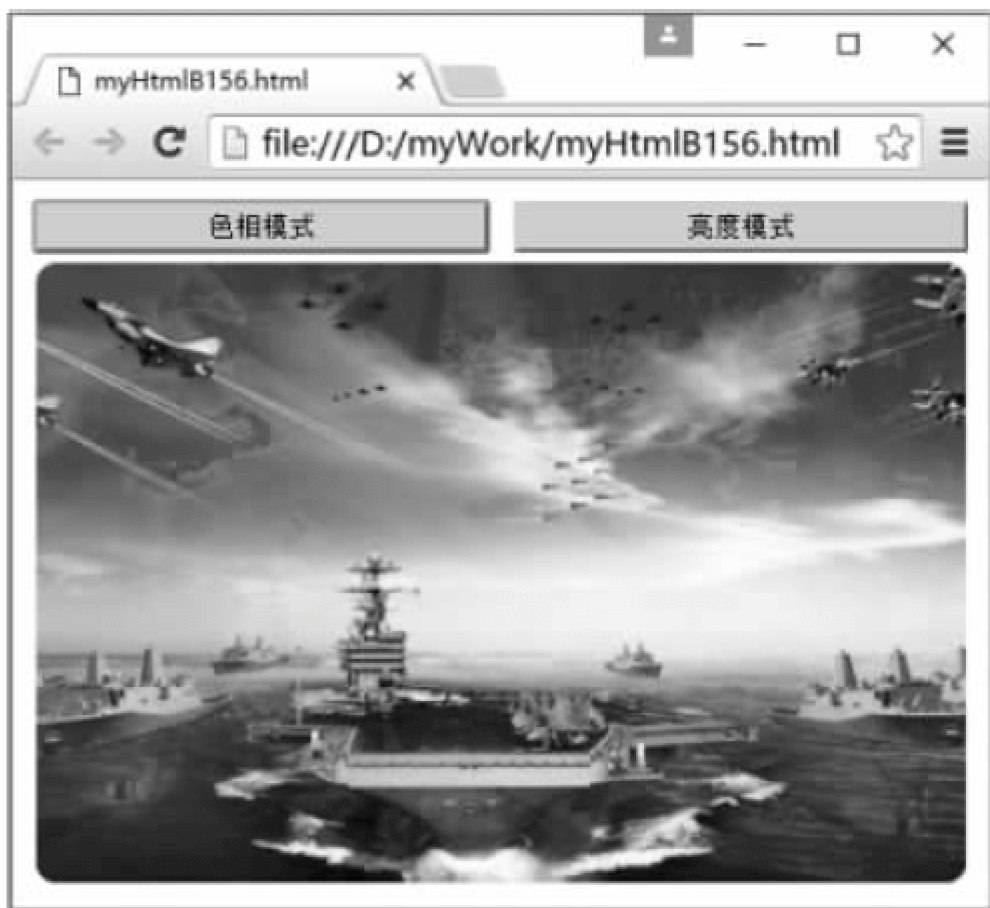


图 239-1

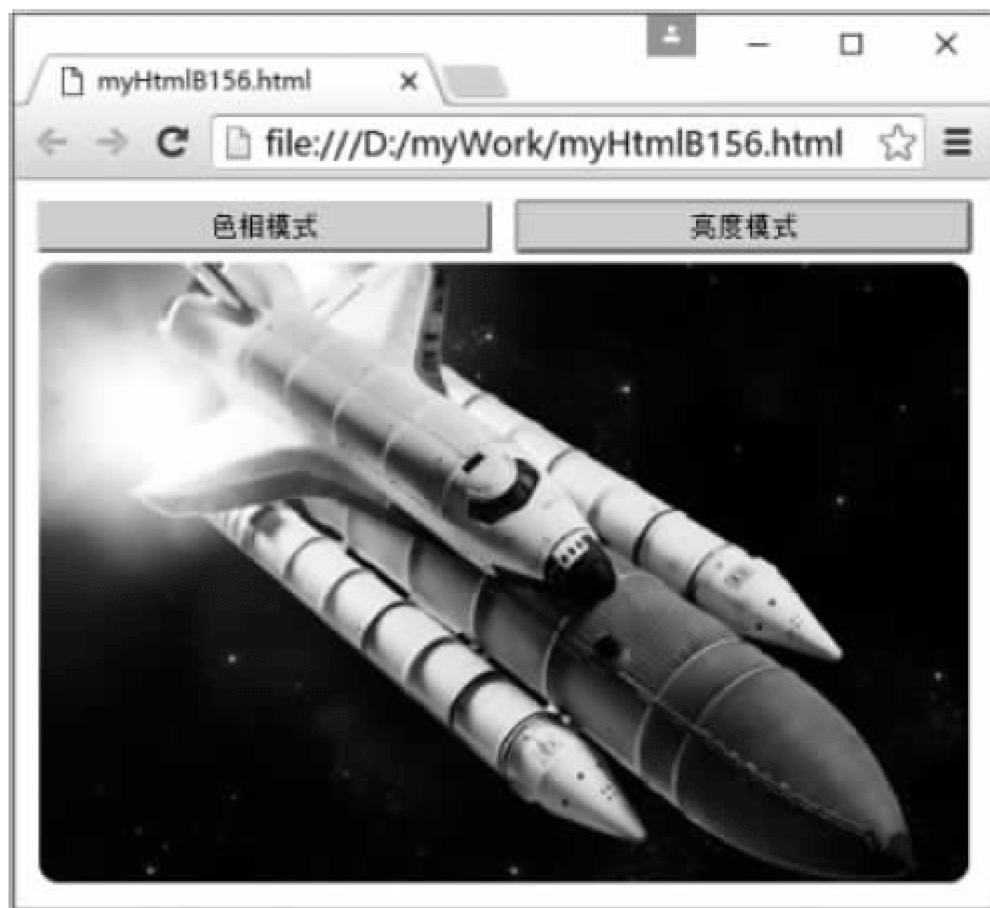


图 239-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnHue").click(function() {//色相模式
    $(".box").css("background-blend-mode", "hue"); });
   $("#myBtnLuminosity").click(function() {//亮度模式
    $(".box").css("background-blend-mode", "luminosity"); });});
 </script>
 <style type="text/css">
  .box {width: 450px;height: 300px;border-radius: 10px;margin: 3px;background: url(img/B156B.jpg)
no-repeat center, url(img/B156A.jpg) no-repeat center;}
  input {width: 220px;border-radius: 2px;padding: 3px;margin: 2px;}
 </style></head>
<body><div><input type="button" value="色相模式" id="myBtnHue"/>
     <input type="button" value="亮度模式" id="myBtnLuminosity"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式（图片与颜色）。$(".box").css("background-blend-mode", "hue")表示元素背景层的两幅图像以色相模式混合显示，色相模式在混合图像时只用混合色颜色的色相值进行着色，而使饱和度和亮度值保持不变。$(".box"). css("background- blend-mode", "luminosity")表示元素背景层的两幅图像以亮度模式混合显示，亮度模式在混合图像时能够使用混合色颜色的亮度值进行着色，而保持基色颜色的饱和度和色相值不变，其实就是用基色中的色相和饱和度以及混合色的亮度创建结果色。

此实例的源文件名是 myHtmlB156.html。

240　以饱和度和颜色模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性分别为 color 和 saturation，从而实现以颜色和饱和度模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击“颜色模式”按钮，则两幅图像经过颜色模式混合后的效果如图 240-1 所示；单击“饱和度模式”按钮，则两幅图像经过饱和度模式混合后的效果如图 240-2 所示。有关此实例的主要代码如下。

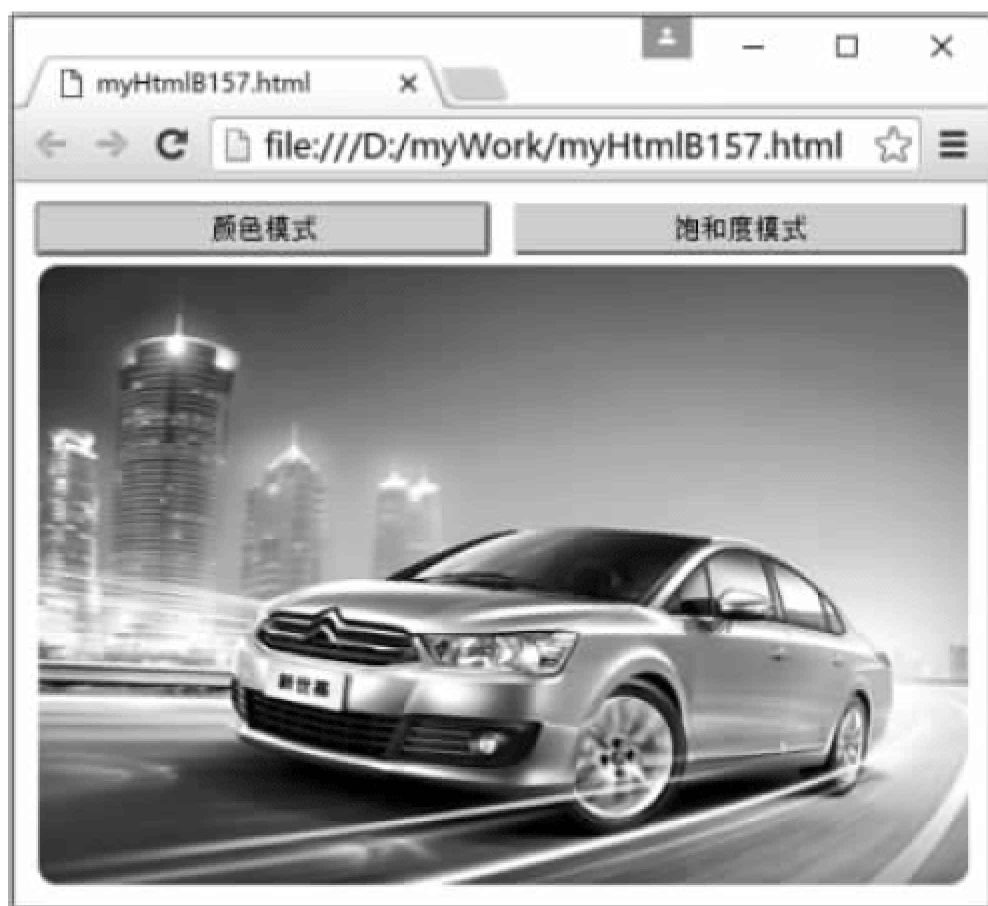


图　240-1



图　240-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnColor").click(function() {//颜色模式
    $(".box").css("background-blend-mode", "color"); });
   $("#myBtnSaturation").click(function() {//饱和度模式
    $(".box").css("background-blend-mode", "saturation"); });});
 </script>
 <style type="text/css">
  .box { width: 450px; height: 300px;border-radius: 10px;margin: 3px;background: url(img/B157A.jpg)
no-repeat center, url(img/B157B.jpg) no-repeat center;}
  input { width: 219px;border-radius: 2px;padding: 3px; margin: 2px;}
 </style></head>
<body><div><input type="button" value="颜色模式" id="myBtnColor"/>
     <input type="button" value="饱和度模式" id="myBtnSaturation"/></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode", "color")表示元素背景层的两幅图像以颜色模式混合显示，颜色模式在混合图像时能够使用混合色的颜色的饱和度值和色相值同时进行着色，而使基色的颜色的亮度值保持不变，颜色模式可以看成是饱和度模式和色相模式的综合。$(".box").css("background -blend-mode", "saturation")表示元素背景层的两幅图像以饱和度模式混合显示，饱和度模式的作用方式与色相模式相似，它只用混合色的颜色的饱和度值进行着色，而使色相值和亮度值保持不变。

此实例的源文件名是 myHtmlB157.html。

241 以正片叠底模式混合显示背景的两幅图像

此实例主要设置元素的 background-blend-mode 属性为 multiply，从而实现以正片叠底模式混合显示背景的两幅图像。当在 Google Chrome 浏览器中显示该页面时，两幅图像在未混合前只显示一幅图像，单击"以正片叠底模式混合显示背景的两幅图像"按钮，则两幅图像经过正片叠底模式混合后的效果如图 241-1 所示。有关此实例的主要代码如下。



图 241-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnMultiply").click(function() {//以正片叠底模式混合显示背景的两幅图像
    $(".box").css("background-blend-mode", "multiply"); });});
 </script><style type="text/css">
  .box {width: 400px;height: 250px;border-radius: 10px;margin: 3px;background: url(img/B158A.jpg)
no-repeat center, url(img/B158B.jpg) no-repeat center;}
  input { width: 400px;border-radius: 2px;padding: 3px;margin: 2px;}
 </style></head>
<body><div><input type="button" value="以正片叠底模式混合显示背景的两幅图像" id=
"myBtnMultiply"/></div><div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式(图片与颜色)。$(".box").css("background-blend-mode","multiply")表示元素背景层的两幅图像以正片叠底模式混合显示，正片叠底模式在混合图像时将查看每个通道的颜色信息，选择基色或混合色复合，结果色通常比源或目标的颜色暗，任何颜色和黑色相乘都得到黑色，任何颜色和白色相乘都会保留原来的颜色。background-blend-mode 属性的默认值是 normal，采用该混合模式的混合色的像素会透过所用的颜色显示出来。

此实例的源文件名是 myHtmlB158.html。

242 以多种混合模式混合显示背景的多幅图像

此实例主要在元素的 background-blend-mode 属性中设置多种混合模式值，从而实现多幅背景图像以多种混合模式处理后的特效显示。当在 Google Chrome 浏览器中显示该页面时，3 幅背景图像在未混合前只显示一幅图像，单击“滤色模式”按钮，则背景图像经过滤色模式混合后的效果如图 242-1 所示；单击“颜色模式”按钮，则背景图像经过颜色模式混合后的效果如图 242-2 所示；单击“滤色模式和颜色模式”按钮，则背景图像经过这两种模式混合后的效果如图 242-3 所示。有关此实例的主要代码如下。



图 242-1

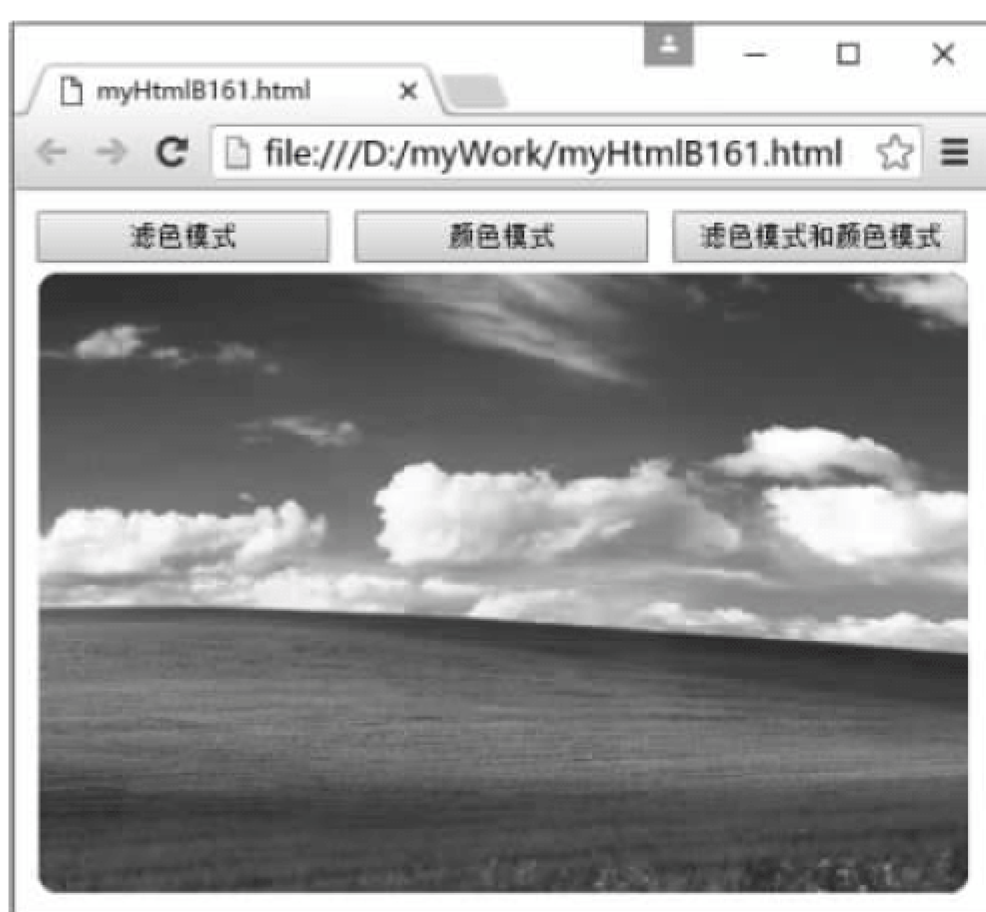


图　242-2



图　242-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnScreen").click(function() {//滤色模式
    $(".box").css("background-blend-mode", "screen"); });
   $("#myBtnColor").click(function() {//颜色模式
    $(".box").css("background-blend-mode", "color"); });
   $("#myBtnMulti").click(function() {//滤色模式和颜色模式
    //$(".box").css("background-blend-mode", "screen,color,hard-light");
    $(".box").css("background-blend-mode", "screen,color"); });});
 </script>
 <style type="text/css">
  .box {width: 450px; height: 300px; border-radius: 10px; margin: 3px;
        background-image: url(img/B153A.jpg),radial-gradient(circle at center, rgb(220, 75, 200),
rgb(0, 0, 75)),url(img/B153B.jpg);}
  button {width: 142px;padding: 3px; margin: 2px;}
 </style></head>
<body><div><button id="myBtnScreen">滤色模式</button>
    <button id="myBtnColor">颜色模式</button>
    <button id="myBtnMulti">滤色模式和颜色模式</button></div>
<div class="box"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-blend-mode 属性用于定义背景层的混合模式（图片与颜色）。$(".box").css("background-blend-mode", "color")表示元素背景层的图像以颜色模式混合显示，颜色模式在混合图像时能够使用混合色的颜色的饱和度值和色相值同时进行着色，而使基色的颜色的亮度值保持不变，颜色模式可以看成是饱和度模式和色相模式的综合效果。$(".box").css("background- blend-mode", "screen")表示元素背景层的图像以滤色模式混合显示，在采用滤色模式混合图像时将查看每个通道的颜色信息，并将混合色的互补色复合，结果总是较亮的颜色，任何颜色和白色进行滤色都得到白色，任何颜色和黑色进行滤色都会保留原来的颜色。$(".box").css("background-blend-mode", "screen,color")表示元素背景层的图像以滤色模式和颜色模式综合处理后显示。需要说明的是，如果 background- blend-mode 属性值有两种混合模式，则 background-image 属性值应该有 3 幅图像；如果 background-blend-mode 属性值有 3 种混合

模式，则 background-image 属性值应该有 4 幅图像，以此类推。background-blend-mode 属性支持的混合模式有 normal、multiply、screen、overlay、darken、lighten、color-dodge、color-burn、hard-light、soft-light、difference、exclusion、hue、saturation、color、luminosity，其中每一种混合模式都有其各自的计算模式。

此实例的源文件名是 myHtmlB161.html。

243 使用镂空技术为 png 背景图标设置颜色

此实例主要使用 png 图标来设置元素的 background-image 属性，并同时设置元素的 background-color 属性，从而实现使用 background-color 属性指定的颜色来显示 background-image 属性指定的 png 图标。当在 Google Chrome 浏览器中显示该页面时，使用红色显示的 png 图标如图 243-1 所示。使用绿色显示的 png 图标如图 243-2 所示，当然可以通过单击“更改图标颜色：”按钮选择不同的颜色来定制不同颜色的图标。有关此实例的主要代码如下。

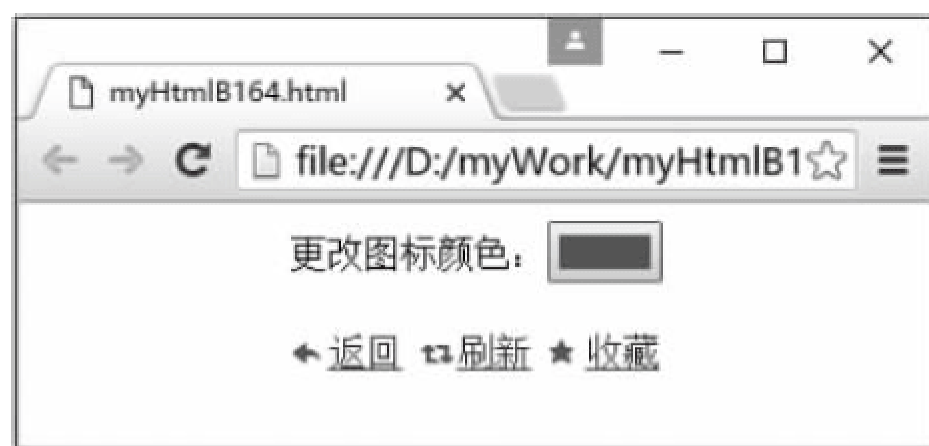


图 243-1

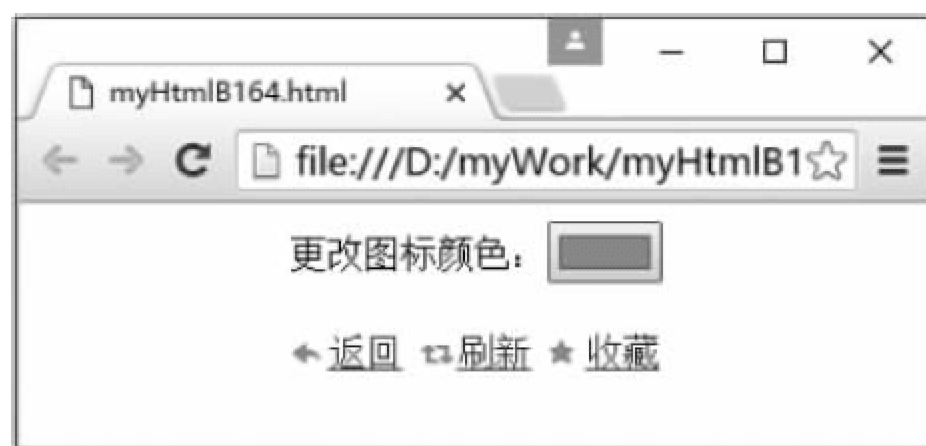


图 243-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myColorDlg").change(function() {
    var myIcons = document.getElementsByTagName("i");
    var i = 0, l = myIcons.length;
    for (; i<l; i += 1) {
      myIcons[i].style.backgroundColor = this.value;
    } });});
 </script>
 <style type="text/css">
  .icon { display: inline-block; width: 16px; height: 20px;
          background-image: url(img/B164.png); background-color: #34538B; }
  .icon1 { background-position: 0 0; }
  .icon2 { background-position: -20px 0;}
  .icon3 { background-position: -40px 0; }
 </style></head>
<body><center>更改图标颜色:
  <input id="myColorDlg" type="color" value="#34538B" autocomplete="off">
  <p><i class="icon icon1"></i><a href="##">返回</a>
      <i class="icon icon2"></i><a href="##">刷新</a>
      <i class="icon icon3"></i><a href="##">收藏</a></p></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myIcons[i].style.backgroundColor=this.value 用于设置每个图标的背景颜色；background-image：url(img/B164.png)用于设置每个图

标的背景图像，由于这种 png 图像不是实心（即镂空）的，所以默认的叠加机制会将 background-color 属性指定的颜色作用于 background-image 属性指定的 png 图标上。需要说明的是，当在 JS 代码中操作 background-color 这种带有“-”符号的属性时，往往需要去掉“-”符号。

此实例的源文件名是 myHtmlB164. html。

244　以绝对定位实现背景模糊、前景清晰的特效

此实例主要设置 position 属性为 absolute 并使用模糊滤镜，从而实现背景模糊、前景清晰的特效。当在 Google Chrome 浏览器中显示该页面时，背景显示的是模糊的图像，文字则以清晰的方式显示，如图 244-1 所示。有关此实例的主要代码如下。



图　244-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBlur { background-image: url(img/B196.jpg); position: absolute;
           width: 400px; height: 250px; -webkit-filter: blur(8px); }
  .myContent { position: absolute; width: 400px;height: 250px;
              font-size: 20px; color: white;text-align: center; }
  img{ width: 50px; height:50px;}
 </style></head>
<body><div class="myBlur"></div>
<div class="myContent"><img src="img/B209A.png">
    <p>曾经沧海难为水,</p><p>除却巫山不是云。</p>
    <p>取次花丛懒回顾,</p><p>半缘修道半缘君。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，position：absolute 表示生成绝对定位的元素，该元素相对于 static（默认值）定位以外的第一个父元素进行定位，因此元素的位置通常应该通过 left、top、right 以及 bottom 属性进行规定；-webkit-filter：blur（8px）表示以 8px 的模糊半径模糊图像。该实例采用了两个 div 元素，一个模糊的 div 元素用于背景，另一个 div 元素用于前景，即将前景 div 元素直接叠加在模糊的 div 元素上（这是因为两个 div 均设置了 position：absolute）。

此实例的源文件名是 myHtmlB251. html。

第3部分 动画

245 使用 transition 属性平滑地旋转图像

此实例主要在CSS样式中设置元素的 transition 属性，从而使图像在指定的时间内旋转指定的角度。当在 Google Chrome 浏览器中显示该页面时，图像在 0°时状态如图 245-1 所示；当鼠标指针悬浮在图像上时，则图像将在 5 秒内以线性过渡的方式从 0°旋转到 270°，如图 245-2 所示。有关此实例的主要代码如下。



图 245-1

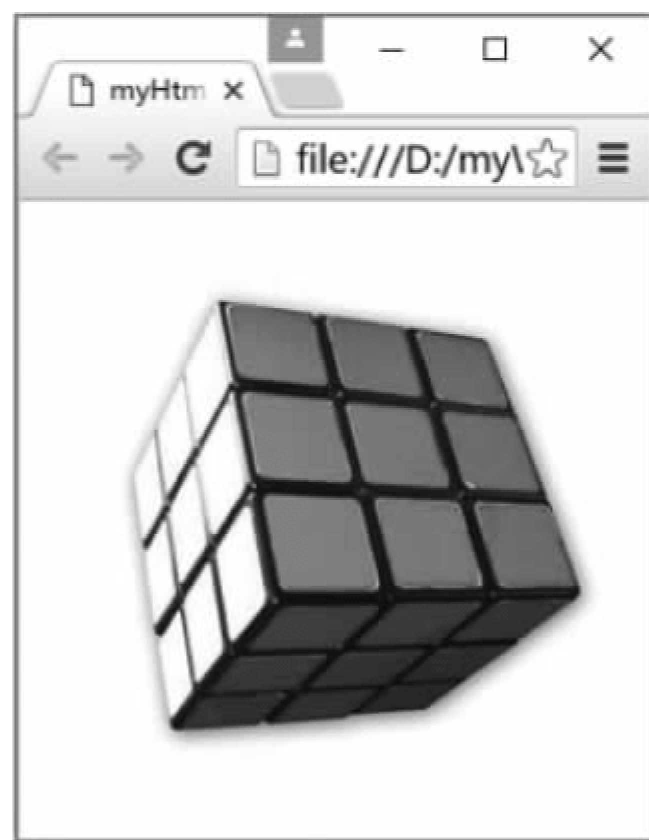


图 245-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  img{ position: absolute; margin: 10px;transition: transform 5s linear;}
  div:hover img{position: absolute;transform: rotate(270deg);}
 </style></head>
<body><div><img src="img/B065.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition: transform 5s linear 用于设置图像在 5 秒内以线性过渡方式执行 transform 所定义的操作(此实例是旋转 270°)。CSS3 的 transition 允许属性值在一定的时间内平滑地过渡，这种效果可以在鼠标单击、获得焦点、被单击或对元素的任何改变中触发，并平滑地以动画效果改变属性值。transition 的语法格式如下。

```
transition: [<'transition - property'> || <'transition - duration'>
|| <'transition - timing - function'> || <'transition - delay'>
[, [<'transition - property'> || <'transition - duration'> ||
<'transition - timing - function'> || <'transition - delay'>]] *
```

从以上可以看出，transition 主要包含 4 个属性值，即 transition-property(执行变换的属性)、transition-duration(变换延续的时间)、transition-timing-function(在延续时间段变换的速率变化)、transition-delay(变换延迟时间)。

transition-property 属性值用来指定当元素的一个属性改变时执行 transition 效果，主要的值为 none(没有属性改变)、all(所有属性改变，这也是其默认值)、ident(元素属性名)。当 ident 值为 none 时，transition 马上停止执行；当 ident 值指定为 all 时，则元素的任何属性值产生变化时都将执行 transition 效果，ident 是可以指定的元素的某一个属性值。ident 对应的类型如下。

(1) color：通过红、绿、蓝和透明度组件变换，例如 border-color、background-color、color、outline-color 等属性。

(2) length：真实的数字，例如 word-spacing、width、vertical-align、top、right、bottom、left、padding、outline-width、margin、min-width、min-height、max-width、max-height、line-height、height、border-width、border-spacing、background-position 等属性。

(3) percentage：真实的数字，例如 word-spacing、width、vertical-align、top、right、bottom、left、min-width、min-height、max-width、max-height、line-height、height、background-position 等属性。

(4) integer：离散步骤(整个数字)，在真实的数字空间以及使用 floor()转换为整数时发生，例如 outline-offset、z-index 等属性。

(5) number：(浮点型)数值，例如 zoom、opacity、font-weight 等属性。

(6) transform list：对元素进行旋转、缩放、移动或倾斜等。

(7) rectangle：通过 x、y、width 和 height(转为数值)变换，例如 crop。

(8) visibility：离散步骤，在 0 到 1 数字范围之内，0 表示隐藏，1 表示完全显示，例如 visibility。

(9) shadow：作用于 color、x、y 和 blur(模糊)属性，例如 text-shadow。

(10) gradient：通过每次停止时的位置和颜色进行变化。它们必须有相同的类型(放射状的或者线性的)和相同的停止数值，以便执行动画，例如 background-image。

(11) paint server(SVG)：只支持从 gradient 到 gradient 以及从 color 到 color，之后的工作与上面类似。

(12) space-separated list of above：如果列表有相同的项目数值，则列表中的每一项按照上面的规则进行变化，否则无变化。

(13) a shorthand property：如果缩写的所有部分都可以实现动画，则像所有单个属性变化一样变化。

transition-duration 属性值用来指定元素转换过程的持续时间，取值< time >为数值，单位为 s(秒)或者 ms(毫秒)，可以作用于所有元素，包括:before 和:after 伪元素。其默认值是 0，也就是变换是即时的。

transition-timing-function 属性值允许根据时间的推进去改变属性值的变换速率，transition-timing-function 有以下 6 个可能值。

(1) ease：逐渐变慢，默认值，ease 函数等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)。

(2) linear：匀速，linear 函数等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)。

(3) ease-in：加速，ease-in 函数等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)。

(4) ease-out：减速，ease-out 函数等同于贝塞尔曲线(0, 0, 0.58, 1.0)。

(5) ease-in-out：加速然后减速，ease-in-out 函数等同于贝塞尔曲线(0.42，0，0.58，1.0)。

(6) cubic-bezier：该值允许自定义一个时间曲线，特定的 cubic-bezier 曲线。(x1，y1，x2，y2)特定于曲线上的点 P1 和点 P2。所有值需要在[0，1]区域内，否则无效。

transition-delay 属性值用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行 transition 效果，其取值< time >为数值，单位为 s(秒)或者 ms(毫秒)，其使用和 transition-duration 极其相似，也可以作用于所有元素，包括 before 和 after 选择器。其默认大小是 0，也就是变换立即执行，没有延迟。

此实例的源文件名是 myHtmlB065.html。

246 使用 transition 属性移动和旋转图像

此实例主要在元素(img)的 transition 属性中设置多个动作属性值，从而使图像在指定的时间内同时实现旋转和移动的动作。当在 Google Chrome 浏览器中显示该页面时，电话机图像的初始状态在左上角；当鼠标指针悬浮在电话机图像上时，则电话机图像立即执行旋转和移动的动作，如图 246-1 所示，并在指定的时间内移动到右下角，如图 246-2 所示。有关此实例的主要代码如下。



图 246-1



图 246-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div {background-image: url("img/a050A.jpg"); width: 400px;height: 250px;}
  img {position: absolute; top: 10px;left: 20px;
       transition: left 5s linear, top 5s linear, transform 3s ease-in-out;}
  div:hover img {position: absolute;left: 290px;top: 140px;
                transform: rotate(360deg);}
 </style></head>
<body><div><img src="img/B066A.png"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition：left 5s linear，top 5s linear，transform 3s ease-in-out 负责完成过渡动画的移动和旋转动作，其中的 transform 是指 rotate(360deg)这个旋转动作，移动操作指元素的 left 值从 20 移动到 290、top 值从 10 移动到 140，linear 表示移动过程是匀速，ease-in-out 表示旋转过程是先加速后减速。

此实例的源文件名是 myHtmlB066.html。

247 使用 transition 属性实现图像的膨胀

此实例主要在 CSS 样式中设置元素的 transition 属性，从而使图像在指定的时间内放大到预定的比例，以产生膨胀特效。当在 Google Chrome 浏览器中显示该页面时，最初的图像如图 247-1 所示；当鼠标指针悬浮在图像上时，则图像将逐渐膨胀，如图 247-2 所示。有关此实例的主要代码如下。



图 247-1



图 247-2

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  div{ width: 400px; height: 250px; overflow: hidden;border - radius: 3px; }
  img {transition: all 3s ease - in - out;}
  img:hover {transform: scale(1.9);}
 </style></head>
<body><div><img src = "img/B196.jpg" /></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition: all 3s ease-in-out 用于设置图像在 3 秒内以 ease-in-out 过渡方式执行所有变更，此处即是 transform 所定义的操作，所以此行代码修改为 transition: transform 3s ease-in-out 也能实现相同的效果；transform: scale(1.9)用于将图像放大 1.9 倍；overflow: hidden 用于实现当放大的图像超出 div 盒子的宽度和高度时自动隐藏超出部分，所以给人的错觉是在膨胀而不是在放大。

此实例的源文件名是 myHtmlB196.html。

248 使用 transition 属性实现侧滑工具栏

此实例主要在 CSS 样式中设置元素的 transition 属性，从而实现在页面右侧布局滑动工具栏按钮。当在 Google Chrome 浏览器中显示该页面时，在页面右侧的绿色工具栏上有 3 个按钮，如果鼠标指针悬浮在第一个按钮上，则将向左滑出说明文字，如图 248-1 所示；如果鼠标指针悬浮在第二个按钮上，也将向左滑出说明文字，如图 248-2 所示；如果鼠标指针悬浮在第三个按钮上，也将向左滑出说明文字。有关此实例的主要代码如下。



图 248-1



图 248-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(function() {
   $('.button').hover(function() {
    $(this).find('p').addClass("button-hover");
   }, function() {
    $(this).find('p').removeClass("button-hover");
  });});
 </script>
 <style type="text/css">
   body { background:url(img/B209D.jpg) ;background-size:cover; }
  /*在页面右侧设置工具栏*/
  .toolbar { position: absolute; right: 0px; top: 0px; width: 35px; height: 100%;
             border-right: 3px solid green; background-color: transparent; }
  /*设置按钮盒子*/
  .myBox { position: absolute;top: 50%; left: 0px;width: 35px; margin-top: -61px;}
  /*设置按钮样式*/
  .button {position: relative; width: 35px; height: 35px; margin-bottom: 1px;
    cursor: pointer;background-color: green; border-radius: 3px 0px 0px 3px;
    font: 12px/150% Arial, Verdana, "宋体";display: inline-block; }
  /*设置按钮左侧的文字*/
   p { width: 92px; height: 35px; line-height: 35px; font-style: normal;
       text-align: center; font-family: "微软雅黑"; position: absolute;
       z-index: 1; left: 35px;top: 0px; background-color: green; color: white;
       border-radius: 3px 0px 0px 3px; margin: 0px; padding: 0px; cursor: pointer;
       transition: left 0.3s ease-in-out;}
  img{ margin: 2px; }
  /*滑动当前按钮左侧的位置*/
  .button-hover { left: -90px; background-color: red;}
 </style></head>
<body><div class="toolbar"><div class="myBox">
 <div class="button"><p>罗曼蒂克消亡史</p>
  <img src="img/B209A.png" width="32" height="32"/></div>
 <div class="button"><p>我不是潘金莲</p>
  <img src="img/B209B.png" width="32" height="32"/></div>
 <div class="button"><p>疯狂动物城</p>
  <img src="img/B209C.png" width="32" height="32"/></div></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition：left 0.3s ease-in-out 用于实现按钮左侧文字在 0.3 秒内以 ease-in-out 过渡方式将 left 属性值从 35px 改变为－90px，背景颜色也设置为红色。

此实例的源文件名是 myHtmlB209.html。

249　使用 transition 属性高仿 toggle 开关

此实例主要在 CSS 样式中设置 checkbox 的 transition 属性，从而实现高仿 toggle 开关的切换效果。当在 Google Chrome 浏览器中显示该页面时，单击 toggle 开关的左端，白色圆圈将从右端滑到左端，并变为黄色圆圈，如图 249-1 所示；单击 toggle 开关的右端，黄色圆圈将从左端滑到右端，并变为白色圆圈，如图 249-2 所示。有关此实例的主要代码如下。



图　249-1

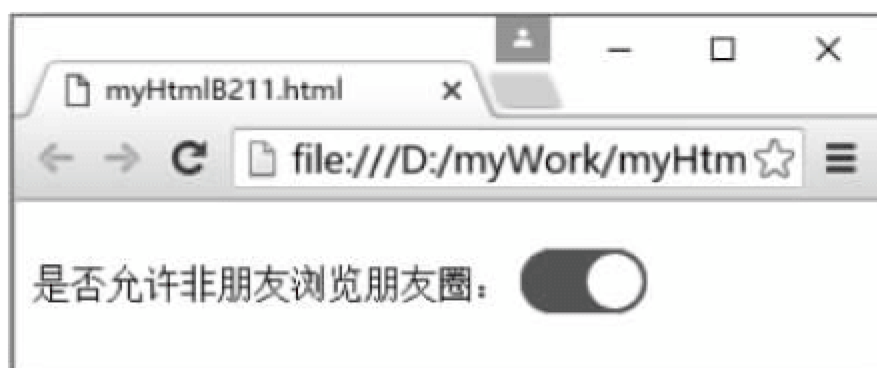


图　249-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*设置左滑槽样式*/
  input[type='checkbox'].toggle { display: inline-block; outline: none;
                                   -webkit-appearance: none;width: 55px; height: 28px; top:10px;
                                  background-color: green; position: relative;
                                   -webkit-border-radius: 28px; -webkit-transition: all 0.2s ease
                                   -in-out;}
  /*设置左滑圆的样式*/
  input[type='checkbox'].toggle:after { content: ''; position: absolute;
                                         display: inline-block; width: 24px; height: 24px; top: 2px;
                                         left: 2px;
                                         background-color: yellow; -webkit-border-radius: 50%;
                                          -webkit-transition: all 0.2s ease-in-out; }
  /*设置右滑槽样式*/
  input[type='checkbox']:checked.toggle { -webkit-box-shadow: inset 0 0 0 15px darkcyan; }
  /*设置右滑圆的样式*/
  input[type='checkbox']:checked.toggle:after { left: 29px;
                                                 background-color: white; }
 </style></head>
<body><div>是否允许非朋友浏览朋友圈：<input class='toggle' type="checkbox" checked='checked'/>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，两个相同的-webkit- transition：all 0.2s ease-in-out 主要用于选中或未选中状态下在 0.2 秒内以 ease-in-out 过渡方式改变圆圈的位置和颜色以及滑槽的颜色。

此实例的源文件名是 myHtmlB211.html。

250　使用 transition 属性旋转菜单指示符

此实例主要在 CSS 样式中设置菜单项右侧的三角形箭头指示符的 transition 属性，从而实现在向上、向下切换指示箭头时出现旋转动画。当在 Google Chrome 浏览器中显示该页面时，“渝北区人民政府”菜单右侧显示的是箭头向下的三角形，使用鼠标单击该菜单项，则将展开该菜单项的二级菜单，同时右侧显示的箭头向下的三角形将逆时针旋转 180°，变为箭头向上的三角形，如图 250-1 所示；使用鼠标再次单击该菜单项，则将折叠该菜单项的二级菜单，同时右侧显示的箭头向上的三角形将顺时针旋转 180°，变为箭头向下的三角形。单击其他菜单项将实现类似的效果。有关此实例的主要代码如下。



图　250-1

```
<!doctype html><html><head><meta charset="UTF-8">
<style type="text/css">
  body { font-family: "Lato", Helvetica, Arial;font-size: 16px; }
  .myBox { width: 350px; }
  /*去掉默认的列表样式*/
  .myBox>ul { list-style: none; }
  /*去掉下画线*/
  .dropdown a { text-decoration: none;}
  /*设置一级菜单的样式*/
  .dropdown [data-toggle="dropdown"] { position: relative; display: block;
                                       color: white;background: lightseagreen; padding: 10px;
                                       border-radius: 3px; margin-bottom: 1px; }
  /*设置一级菜单的悬浮颜色*/
   .dropdown [data-toggle="dropdown"]:hover {background: darkblue;}
   /*设置右侧的三角形指示符的样式*/
   .dropdown .icon-arrow { position: absolute; display: block; font-size: 18px;
                           color: #FFF;top: 10px;right: 10px; }
  /*反转三角形指示符*/
```

```
    .dropdown .icon-arrow.open { -webkit-transform: rotate(-180deg);
                                 -webkit-transition: -webkit-transform 1s;}
    /*三角形指示符复位*/
    .dropdown .icon-arrow.close { -webkit-transform: rotate(0deg);
                                  -webkit-transition: -webkit-transform 1s;}
    /*设置三角形指示符*/
    .dropdown .icon-arrow:before { content: '\25BC'; }
    /*设置二级菜单*/
    .dropdown .dropdown-menu { max-height: 0; overflow: hidden;
                               list-style: none; padding: 0;margin: 0; font-size: 14px; }
    /*设置二级菜单项*/
    .dropdown .dropdown-menu li a { display: block; color: black; background: cyan;
                                    padding: 10px 10px; margin-bottom: 1px; border-radius: 3px;}
    /*设置二级菜单项的悬浮颜色 */
    .dropdown .dropdown-menu li a:hover { background: red; }
    .dropdown .show, .dropdown .hide { -webkit-transform-origin: 50% 0%; }
    /*显示二级菜单*/
    .dropdown .show { display: block; max-height: 9999px;
                      -webkit-transform: scaleY(1); -webkit-transition: max-height 0.1s ease-in-
                      out; }
    /*隐藏二级菜单*/
    .dropdown .hide { max-height: 0; -webkit-transform: scaleY(0);
                      -webkit-transition: max-height 0.1s ease-out; }
  </style></head>
<body><div class="myBox"><ul><li class="dropdown">
  <a href="#" data-toggle="dropdown">渝北区人民政府<i class="icon-arrow"></i></a>
  <ul class="dropdown-menu">
    <li><a href="#">渝北区民政局</a></li><li><a href="#">渝北区财政局</a></li>
    <li><a href="#">龙山街道办事处</a></li><li><a href="#">回兴街道办事处</a>
</li>
    <li><a href="#">双龙街道办事处</a></li></ul></li>
  <li class="dropdown">
  <a href="#" data-toggle="dropdown">长寿区人民政府<i class="icon-arrow"></i></a>
  <ul class="dropdown-menu">
    <li><a href="#">长寿区环保局</a></li><li><a href="#">长寿区移民局</a></li>
    <li><a href="#">凤城街道办事处</a></li><li><a href="#">江南街道办事处</a>
</li></ul></li>
    <li class="dropdown">
  <a href="#" data-toggle="dropdown">江北区人民政府<i class="icon-arrow"></i></a>
    <ul class="dropdown-menu">
    <li><a href="#">江北区审计局</a></li><li><a href="#">五里店街道办事处</a>
</li>
    <li><a href="#">观音桥街道办事处</a></li></ul></li></ul></div>
<script>
var dropdown = document.querySelectorAll('.dropdown');
var dropdownArray = Array.prototype.slice.call(dropdown, 0);
dropdownArray.forEach(function(el) {
  var button = el.querySelector('a[data-toggle="dropdown"]'),
      menu = el.querySelector('.dropdown-menu'),
      arrow = button.querySelector('i.icon-arrow');
  button.onclick = function(event) {
      if (!menu.hasClass('show')) {
       menu.classList.add('show'); menu.classList.remove('hide');
```

```
            arrow.classList.add('open'); arrow.classList.remove('close');
            event.preventDefault();
           } else {
            menu.classList.remove('show'); menu.classList.add('hide');
            arrow.classList.remove('open'); arrow.classList.add('close');
            event.preventDefault();
     } };})
     Element.prototype.hasClass = function(className) {
      return this.className && new RegExp("(^|\\s)" + className + "(\\s| $ )").test(this.className); };
</script></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，两个相同的-webkit- transition:-webkit-transform 1s 主要用于单击一级菜单时在 1 秒内以顺时针或逆时针的方式旋转右侧的三角形指示符。实际上，以顺时针或逆时针的方式旋转右侧的三角形指示符是通过-webkit-transform：rotate(0deg)或-webkit-transform：rotate(－180deg)代码完成的，如果没有-webkit-transition：-webkit-transform 1s 这行代码，用户就看不见三角形箭头指示符的上下切换过程，没有那种动感炫酷的效果。

此实例的源文件名是 myHtmlB212.html。

251 使用 transition 属性高仿纸张卷边

此实例主要通过 transition 属性动态绘制渐变色背景，从而实现动态变化纸张边角卷边的特效。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在图像上，则图像的左上角开始卷边，如图 251-1 所示。有关此实例的主要代码如下。



图 251-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <style type = "text/css">
   .myImg { margin: .4em; padding: 1em; background: #ECECEC;
            color: #666; border - radius: 2px; }
   .curl - top - left { display: inline - block; position: relative; }
```

```
    .curl - top - left:before { position: absolute; content: '';
         height: 0px; width: 0px; top: 0; left: 0; background:
          linear - gradient(135deg, white 45 % , ＃AAA 50 % , ＃CCC 56 % , white 80 % );
         transition: all 1s ease - in - out }
    /＊设置卷角大小＊/
    .curl - top - left:hover:before { width: 45px; height: 45px; }
  </style></head>
 <body><center><a class = "myImg curl - top - left"><img src = "img/B222.jpg"></a>
 </center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition: all 1s ease-in-out 主要用于在鼠标指针悬浮于图像或离开图像时展开或收缩卷边的渐变背景，即卷边的这种特效本质上就是一幅大小通过过渡动画进行变化的渐变背景；background: linear-gradient(135deg, white 45%, ＃AAA 50%, ＃CCC 56%, white 80%)用于根据预置的宽度和高度按照比例绘制渐变背景。

此实例的源文件名是 myHtmlB222.html。

252　使用 transition 属性实现悬空阴影

此实例主要通过 transition 属性动态改变椭圆阴影的透明度，从而实现为图像创建悬空的阴影。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在图像上，则图像将悬空在椭圆阴影的上方，如图 252-1 所示。有关此实例的主要代码如下。



图　252-1

```
 <!doctype html><html><head><meta charset = UTF - 8>
  <style type = "text/css">
   .myShadow { margin: .4em; padding: 1em; }
   .hover - shadow { display: inline - block; position: relative; }
   .hover - shadow:before { pointer - events: none; position: absolute;
        content: ''; top: 100 % ; left: 5 % ; height: 20px; width: 85 % ; opacity: 0;
        background: radial - gradient(ellipse at center, rgba(0, 0, 0, 0.35) 0 % , rgba(0, 0, 0, 0) 80 % );
        transition: opacity 0.3s ease - in - out; }
   .hover - shadow:hover:before { opacity: 0.8; }
  </style></head>
 <body><center><a class = "myShadow hover - shadow"><img src = "img/B209A.png"></a>
 </center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition：opacity 0.3s ease-in-out 主要用于在鼠标指针悬浮于图像或离开图像时通过改变透明度(0 至 0.8)逐渐显示或隐藏椭圆形的渐变背景，opacity 属性值为 0.0 表示完全透明，opacity 属性值为 1.0 表示完全不透明；background：radial-gradient(ellipse at center，rgba(0，0，0，0.35) 0%，rgba(0，0，0，0) 80%)表示在图像下方的水平中心位置创建渐变透明(0.35 至 0)的放射椭圆。

此实例的源文件名是 myHtmlB223.html。

253　使用 transition 属性实现纸张卷拱

此实例主要通过 transition 属性动态改变两个中心渐变椭圆的透明度，从而实现为超链接创建纸张卷拱的阴影特效。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针未放在超链接“炫酷应用实例集锦”上，则显示正常的超链接；如果鼠标指针悬浮在超链接“炫酷应用实例集锦”上，则在超链接的上、下方将出现中心渐变阴影，如同纸张卷拱的效果，如图 253-1 所示。有关此实例的主要代码如下。



图　253-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style>
  .myShow { margin: .4em; padding: 1em; cursor: pointer; background: #ECECEC;
          text-decoration: none;color: #666; width: 200px; }
  .shadow-radial { display: inline-block; position: relative; }
  .shadow-radial:before, .shadow-radial:after { position: absolute; content: '';
           left: 0; width: 100%; height: 5px; opacity: 0;
           /*改变透明度显示或隐藏渐变阴影*/
           transition: opacity 0.5s ease-in-out; }
  /*绘制上面的渐变阴影*/
  .shadow-radial:before { bottom: 100%; background: radial-gradient(ellipse at 50% 150%, rgba(0,
                      0, 0, 0.6) 0%, rgba(0, 0, 0, 0) 80%); }
  /*绘制下面的渐变阴影*/
  .shadow-radial:after { top: 100%;background: radial-gradient(ellipse at 50% -50%, rgba(0, 0, 0,
                     0.6) 0%, rgba(0, 0, 0, 0) 80%); }
  /*显示渐变阴影*/
  .shadow-radial:hover:before, .shadow-radial:hover:after { opacity: 1;}
 </style></head>
<body><center><a class = "myShow shadow-radial">炫酷应用实例集锦</a></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transition：opacity 0.5s ease-in-out 主要用于在鼠标指针悬浮于超链接或离开超链接时通过改变透明度(0 至 1)逐渐显示或隐藏超链接上、下的两个中心渐变阴影，opacity 属性值为 0.0 表示完全透明，opacity 属性值为 1.0 表示完全不透明；background：radial-gradient(ellipse at 50% 150%，rgba(0，0，0，0.6) 0%，rgba(0，0，0，0) 80%)表示在图像上方的水平中心位置创建渐变透明(0.6 至 0)的中心放射椭圆。

此实例的源文件名是 myHtmlB227.html。

254 使用 transition 属性实现图像由模糊变清晰

此实例主要使用 transition 属性动态改变模糊滤镜-webkit-filter，从而实现图像由模糊状态逐渐变为清晰状态。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针未悬浮在图像上，则显示模糊的图像，如图 254-1 所示；如果鼠标指针悬浮在图像上，则图像逐渐变得清晰，直到显示原图。有关此实例的主要代码如下。

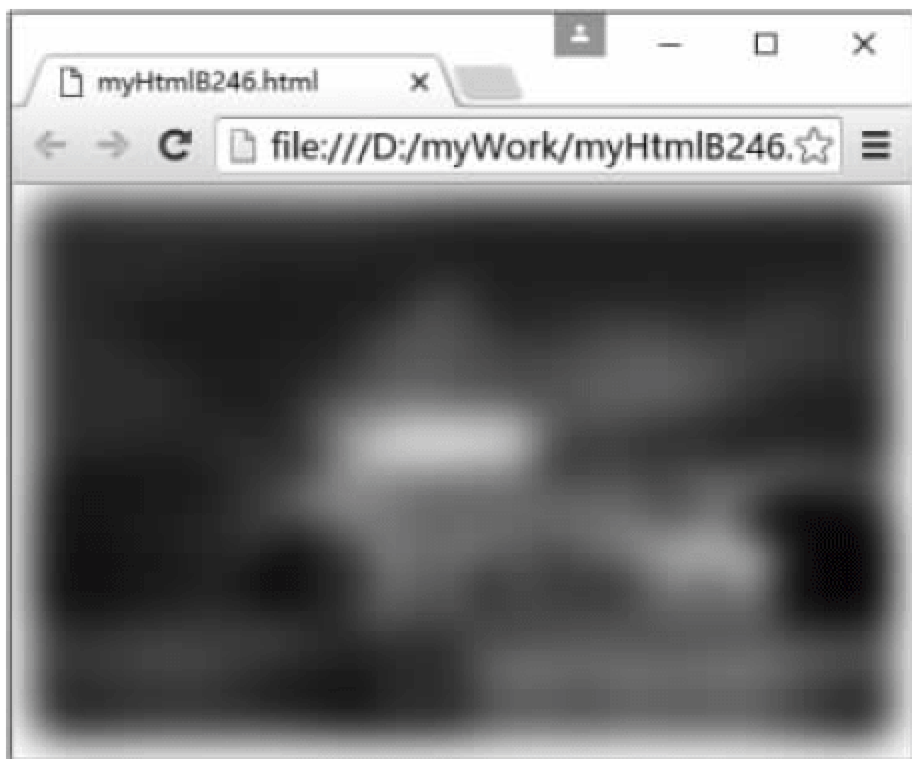


图 254-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   img { width:400px; height:250px; border-radius: 10px;
         /* 显示模糊半径为 10px 的图像 */
         -webkit-filter: blur(10px); -webkit-transition: all 2s ease-in-out; }
   img:hover { /* 在鼠标指针悬浮时显示清晰图像 */
              -webkit-filter: blur(0px); }
 </style></head>
<body><center><img src="img/B246.jpg"></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-filter：blur(0px)表示滤镜的模糊半径为 0px，即一点也不模糊；-webkit-filter：blur(10px)表示滤镜的模糊半径为 10px，即相当模糊；-webkit-transition：all 2s ease-in-out 表示在两秒内以 ease-in-out 方式将滤镜的模糊半径由 10px 改变为 0px，该代码在此实例中也等于-webkit- transition：-webkit-filter 2s ease-in-out。

此实例的源文件名是 myHtmlB246.html。

255 使用 transition 属性实现动态拉伸文本框的边线

此实例主要通过使用 transition 属性动态改变 span 元素的宽度，从而实现动态拉伸焦点文本框底部边线的特殊效果。当在 Google Chrome 浏览器中显示该页面时，如果"申请职位："文本框获得焦点，则其灰色底线上立即有一条蓝色的线条从左向右拉伸，直到完全覆盖灰色底线，此时即可在"申请职位："文本框中输入信息；如果"毕业院校："文本框获得焦点，则"申请职位："文本框的蓝色底线将从右向左收缩，直到露出全部的灰色底线，此时"毕业院校："文本框的灰色底线上立即有一条蓝色的线条从左向右拉伸，直到完全覆盖原来的灰色底线，然后即可在"毕业院校："文本框中输入信息，如图

255-1 所示；如果"职称名称："文本框获得焦点，则"毕业院校："文本框的蓝色底线将从右向左收缩，直到露出全部的灰色底线，此时"职称名称："文本框的灰色底线上立即有一条蓝色的线条从左向右拉伸，直到完全覆盖原来的灰色底线，然后即可在"职称名称："文本框中输入信息。有关此实例的主要代码如下。



图 255-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*隐藏获取焦点时的边框线*/
  :focus { outline: none; }
  /*创建盒子模型*/
  .myBox { width: 300px; margin-top: 30px; position: relative; }
  /*设置输入框的大小*/
  input[type="text"] { font: 15px/24px "Lato", Arial, sans-serif;width: 190px; }
  /*隐藏除底线外的其他 3 条边框线*/
  .myEffect { border: 0; border-bottom: 1px solid #CCC; }
  /*设置过渡动画的位置、距离、时间、颜色*/
  .myEffect ~ .focus-border { position: absolute; bottom: 0;left: 100px;
                              width: 0; height: 2px;background-color: #3399FF;
                              transition: 0.4s;}
  /*设置过渡动画的距离(从 0 增加到 190,然后反向执行)*/
  .myEffect:focus ~ .focus-border { width: 190px;}
 </style></head>
<body><center>
 <div class="myBox">申请职位:<input class="myEffect" type="text" placeholder="请参考招聘信息"/>
<span class="focus-border"></span></div>
 <div class="myBox">毕业院校:<input class="myEffect" type="text" placeholder="教育部认可的高等
院校"/><span class="focus-border"></span></div>
 <div class="myBox">职称名称:<input class="myEffect" type="text" placeholder="职业资格证书"/>
<span class="focus-border"></span></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，position：absolute 表示元素脱离正常文档流，使用 top、right、bottom、left 等属性进行绝对定位，而其层叠通过 z-index 属性定义，在此实例中因为 span 是直接叠加在 input 上，所以需要用 left：100px 以绝对定位方式进行重叠；transition：0.4s 表示在 0.4 秒内完成动画，由于此动画只是将线条的长度进行拉伸和缩减，所以此代码也可写成 transition：0.4s width。

此实例的源文件名是 myHtmlB252.html。

256 使用 transition 属性实现从边线两端向中心靠拢

此实例主要使用 transition 属性动态改变 span 元素的宽度，并将其伸缩行为分解在 before 选择器和 after 选择器中，从而实现从边线两端向中心靠拢的特殊动态效果。当在 Google Chrome 浏览器中显示该页面时，如果“申请职位：”文本框获得焦点，则其灰色底线上立即有两条蓝色的线条分别从左、右两端向中心靠拢，直到完全覆盖灰色底线，此时即可在“申请职位：”文本框中输入信息；如果“毕业院校：”文本框获得焦点，则“申请职位：”文本框的蓝色底线将从中心分别向左、右两端收缩，直到露出全部的灰色底线，此时“毕业院校：”文本框的灰色底线上立即有两条蓝色的线条分别从左、右两端向中心靠拢，直到完全覆盖原来的灰色底线，然后即可在“毕业院校：”文本框中输入信息，如图 256-1 所示；如果“职称名称：”文本框获得焦点，则“毕业院校：”文本框的蓝色底线将从中心分别向左、右两端收缩，直到露出全部的灰色底线，此时“职称名称：”文本框的灰色底线上立即有两条蓝色的线条分别从左、右两端向中心靠拢，直到完全覆盖原来的灰色底线，然后即可在“职称名称：”文本框中输入信息。有关此实例的主要代码如下。

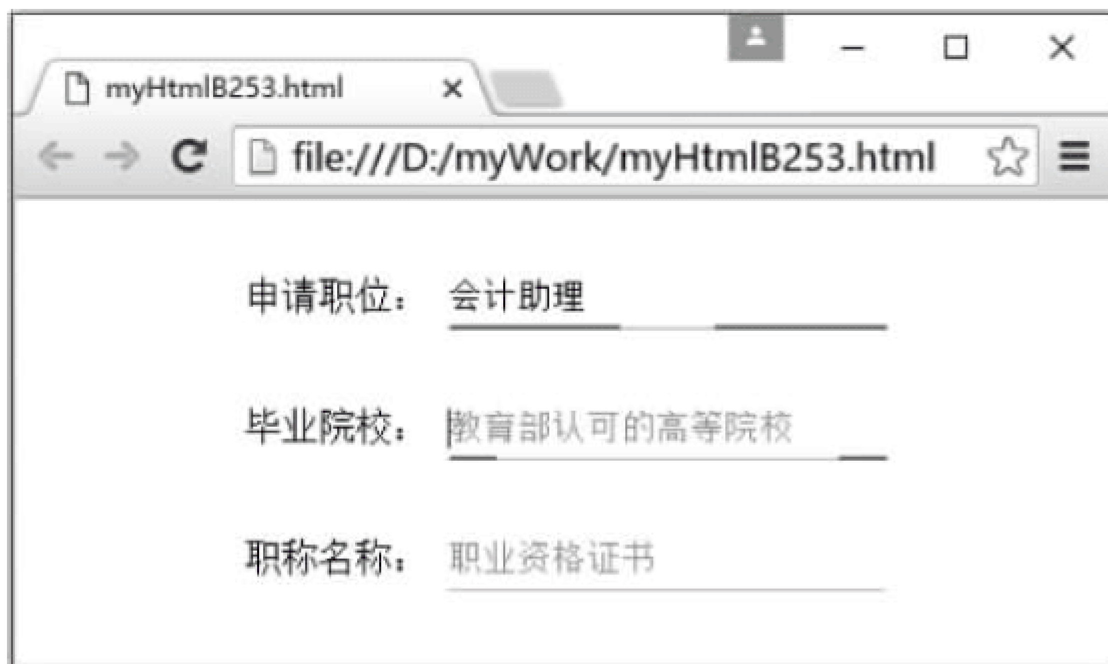


图 256-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*隐藏获取焦点时的边框线*/
  :focus { outline: none; }
  /*创建盒子模型*/
  .myBox { width: 300px; margin-top: 30px; position: relative; }
  /*设置输入框的大小*/
  input[type="text"] { font: 15px/24px "Lato", Arial, sans-serif;width: 190px; }
  /*隐藏除底线外的其他3条边框线*/
  .myEffect { border: 0; border-bottom: 1px solid #CCC; }
  /*设置演示收缩、拉伸的底线*/
  .myEffect ~ .focus-border { position: absolute; bottom: 0;
                              left: 100px; width: 190px; height: 2px; z-index: 99; }
  .myEffect ~ .focus-border:before, .myEffect ~ .focus-border:after {
            content: ""; position: absolute; bottom: 0; left: 0px;
            /*设置左、右初始宽度为0*/
            width: 0; height: 100%;background-color: #3399FF; transition: 2s;}
   /*负责右半部分拉伸,表示离右端的距离为0,即向右收缩*/
   .myEffect ~ .focus-border:after { left: auto; right: 0; }
```

```
    /＊左、右两部分各完成宽度的一半＊/
    .myEffect:focus ～ .focus-border:before,
         .myEffect:focus ～ .focus-border:after { width: 50%; }
  </style></head>
<body><center><div class="myBox">申请职位:<input class="myEffect" type="text" placeholder="请
参考招聘信息"/><span class="focus-border"></span></div>
 <div class="myBox">毕业院校:<input class="myEffect" type="text" placeholder="教育部认可的高等
院校"/><span class="focus-border"></span></div>
 <div class="myBox">职称名称:<input class="myEffect" type="text" placeholder="职业资格证书"/>
<span class="focus-border"></span></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中 position: absolute 表示对元素进行绝对定位,在此实例中因为 span 是直接叠加在 input 上,所以需要用 left: 100px 以绝对定位方式进行重叠; transition: 2s 表示在两秒内同时完成左、右两个伸缩动画,由于此动画只是将线条的长度进行拉伸和缩减,所以此代码也可写成 transition: 2s width; .myEffect ～ .focus-border:before 表示在 myEffect ～ .focus-border 的前面插入内容; .myEffect ～ .focus-border:after 表示在.myEffect ～ .focus-border 的后面插入内容。

此实例的源文件名是 myHtmlB253.html。

257 使用 transition 属性实现动态滑出焦点按钮的背景

此实例主要通过使用 transition 属性动态改变焦点按钮背景的宽度,从而实现动态滑出焦点按钮背景的特殊效果。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在“清华大学出版社”按钮上,则该按钮将从左向右滑出浅绿色的背景;如果鼠标指针悬浮在“中国水利水电出版社”按钮上,则该按钮将从左向右滑出浅绿色的背景,同时“清华大学出版社”按钮的浅绿色背景将从右向左收缩,直到完全消失,如图 257-1 所示;如果鼠标指针悬浮在“电子科技大学出版社”按钮上将实现类似的效果。有关此实例的主要代码如下。



图 257-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /＊设置盒子＊/
  .myBox { height: 37px; width: 250px; margin-top: 5px;
          background-color: #FEB41D; border-radius: 5px;}
   /＊设置盒子中的按钮＊/
  .myButton {line-height: 37px; height: 37px; width: 250px;
            text-align: center; margin-right: auto; margin-left: auto;
            cursor: pointer; border-radius: 5px;position: relative; }
```

```
    /* 设置按钮文字 */
    .myButton span { display: block; position: absolute;
                     width: 100%; height: 100%; color:black; }
    .myButton::before { border-radius: 5px; content: ''; position: absolute;
                        top: 0; left: 0; /* 设置动画起点 */
                        width: 0%; height: 100%;background-color:lightgreen;
                        /* 设置动画时间 */
                        -webkit-transition: all 0.3s; }
    .myButton:hover::before { width: 100%; }
  </style></head>
<body><center><div class="myBox"><div class="myButton"><span>清华大学出版社</span></div></div>
  <div class="myBox"><div class="myButton"><span>中国水利水电出版社</span></div> </div>
  <div class="myBox"><div class="myButton"><span>电子科技大学出版社</span></div> </div></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，position: absolute 表示对元素进行绝对定位，在一般情况下，如果元素需要使用 left、top、right、bottom 进行绝对定位，则需要先设置 position: absolute; -webkit-transition: all 0.3s 表示在 0.3 秒内从左向右滑出浅绿色的背景或从右向左收缩浅绿色的背景；.myButton::before 表示在.myButton 的前面插入内容，在此实例中.myButton::before 也可以写成.myButton:before。

此实例的源文件名是 myHtmlB254.html。

258 使用 transition 属性高仿扑克牌正、反面的旋转

此实例主要使用 transition 属性改变元素在 Z 轴的堆叠顺序和 Y 轴的旋转角度，从而实现高仿扑克牌正、反面的旋转切换的动态效果。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在扑克牌正面上，如图 258-1 所示，则扑克牌的正面将围绕 Y 轴旋转 180°，同时扑克牌的反面将反向围绕 Y 轴旋转 180°，如图 258-2 所示；如果鼠标指针悬浮在扑克牌反面上，如图 258-2 所示，则扑克牌的反面将围绕 Y 轴旋转－180°，同时扑克牌的正面将反向围绕 Y 轴旋转－180°，如图 258-1 所示。有关此实例的主要代码如下。



图 258-1



图 258-2

```
<!doctype html><html><head><meta charset="UTF-8">
  <style type="text/css">
    /* 设置扑克牌盒子 */
```

```
    .myBox { position: absolute; margin: 30px 35 % ; }
    /*旋转后反面在前*/
    .myBox:hover .back { -webkit-transform: rotateY(0deg);z-index: 2; }
    /*旋转后正面在后*/
    .myBox:hover .front { -webkit-transform: rotateY(180deg);z-index: 1; }
    /*正面在前*/
    .front { z-index: 2; -webkit-transform: rotateY(0deg); }
    /*反面在后*/
    .back { z-index: 1; -webkit-transform: rotateY(-180deg);}
    /*设置正、反面图像的大小*/
    img { width: 108px; height: 150px;overflow: hidden; border-radius: 5px; }
    /*设置过渡动画属性*/
    .front, .back { position: absolute;top: 0; left: 0;
                    -webkit-transition: 1s ease-out; }
  </style></head>
<body><div class="myBox "><div class="front"><img src="img/B256A.jpg"/></div>
  <div class="back"><img src="img/B256B.jpg"/></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，position：absolute 表示对元素进行绝对定位，在一般情况下，如果元素需要使用 left、top、right、bottom 进行绝对定位，则需要先设置 position：absolute，因为在此实例中扑克牌的正、反面是两个独立的 div，它们根据 z-index 属性值解决重叠问题，所以旋转之后的最后一瞬间需要进行绝对定位；rotateY(180deg)表示将扑克牌从当前角度围绕 Y 轴旋转 180°；z-index 属性用于设置扑克牌正、反面的堆叠顺序，拥有更高堆叠顺序(z-index 属性值大小)的扑克牌正(反)面总是会处于堆叠顺序(z-index 属性值大小)的扑克牌反(正)面的前面；-webkit-transition：1s ease-out 表示在 1 秒内以 ease-out 模式围绕 Y 轴旋转 180°，同时更改 z-index 属性值，需要特别注意的是，此时扑克牌的正、反面两个过渡动画是在同时反向旋转和更改 z-index 属性值，即是两个动画，而不是一个动画在执行。

此实例的源文件名是 myHtmlB256.html。

259 使用 transition 和 transform 属性实现平行四边形风格菜单

此实例主要通过综合使用 transition 和 transform 属性实现平行四边形风格的菜单。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在“集团首页”菜单上，显示的平行四边形效果如图 259-1 所示；如果鼠标指针悬浮在“组织架构”菜单上，显示的平行四边形效果如图 259-2 所示。当然，鼠标指针悬浮在其他菜单上也能实现类似的效果。有关此实例的主要代码如下。



图 259-1



图 259-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  *{ margin: 0; padding: 0; list-style: none; }
  body{ background-color: snow; }
  ul { margin-left: 12px; }
  li { float: left; }
  /*设置菜单盒子样式*/
  .myBox { margin-top: 50px; width: 100%; height: 35px;
          background-color: blue;box-shadow: 2px 2px 8px black; }
  /*设置菜单项样式*/
  .myItem { padding-top: 8px; width: 100px;height: 27px; text-align: center;
           -webkit-transform: skew(-25deg); -webkit-transition: all 1s ease;
           font-size: 16px; font-weight: bold; color: white; }
  /*设置悬浮菜单项样式*/
  .myItem:hover { background-color: red; }
 </style></head>
<body><div class="myBox"><ul><li><div class="myItem">集团首页</div></li>
     <li><div class="myItem">新闻中心</div></li>
     <li><div class="myItem">组织架构</div></li>
     <li><div class="myItem">联系我们</div></li></ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：skew(－25deg)用于使菜单项向右倾斜25°，-webkit-transform：skew(25deg)则使菜单项向左倾斜25°，用户可以根据需要进行选择，因为默认的div元素是矩形，当它倾斜25°之后自然变成一个平行四边形；-webkit-transition：all 1s ease表示在1秒内以ease模式将焦点菜单项的背景置为红色或从红色恢复为原来的背景色。

此实例的源文件名是myHtmlB257.html。

260　使用transition属性和translateY()方法实现在图像上滑出简介层

此实例主要通过综合使用transition属性和translateY()方法实现当鼠标指针悬浮在图像上时从下向上滑出图像简介的效果。当在Google Chrome浏览器中显示该页面时，如果鼠标指针悬浮在第一幅图像上，则从下向上滑出该图像的简介，如图260-1所示；如果鼠标指针悬浮在第二幅图像上，也将从下向上滑出该图像的简介，如图260-2所示；如果鼠标指针悬浮在第三幅图像上，仍将从下向上滑出该图像的简介。有关此实例的主要代码如下。



图　260-1



图　260-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  *{ margin: 0;padding: 0;}
  body{ background-color: black;}
  /*设置盒子样式*/
  .myBox { display: inline-block; padding: 0px;border-radius: 5px; margin: 10px; }
  /*设置文字背景层样式*/
  [class^='myBrief-'], [class*='myBrief-'] {border-radius: 5px;
              position: relative; color: yellow; display: inline-block;
              background-color: rgba(1, 87, 155, 0.75); overflow: hidden; }
  /*设置文字位置*/
  [class^='myBrief-'] div, [class*='myBrief-'] div { background-color: inherit;
        padding: 5px; position: absolute; top: 0; bottom: 0; left: 0; right: 0; }
  p{ text-align: left; margin: 5px; font-size: 14px; }
  h3{ margin: 5px; }
  /*平移动画*/
  [class^='myBrief-slide-']:hover div, [class*='myBrief-slide-']:hover div {
        -webkit-transform: translate(0); -webkit-transition: all 0.35s ease; }
  .myBrief-slide-up div { -webkit-transform: translateY(100%); }/*垂直平移*/
 </style></head>
<body><center><div class="myBox">
  <div class="myBrief-slide-up"><img src="img/B259A.jpg" >
   <div><h3>镇远古镇</h3><p>镇远古镇是贵州省黔东南苗族侗族自治州镇远县名镇,位于舞阳河畔,四周
皆山。河水蜿蜒,以S形穿城而过,北岸为旧府城,南岸为旧卫城,远观颇似太极图。两城池皆为明代所建,现尚
存部分城墙和城门。</p></div></div></div>
 <div class="myBox"><div class="myBrief-slide-up"><img src="img/B259B.jpg" >
  <div><h3>丽江古镇</h3><p>丽江古城,又名"大研古镇",世界文化遗产,国家5A级旅游景区,全国文明风
景旅游区示范点。位于中国西南部云南省的丽江市,坐落在丽江坝中部,玉龙雪山下,北倚象山、金虹山、西枕狮
子山。</p></div></div></div>
 <div class="myBox"><div class="myBrief-slide-up"><img src="img/B259C.jpg" >
  <div><h3>西塘古镇</h3><p>西塘是江南六大古镇之一,位于浙江省嘉兴市嘉善县。嘉善位于上海西南方
向与上海零距离接壤,距上海市中心80公里,大虹桥商务区60公里,西至杭州110公里,南濒嘉兴港乍浦港区
35公里,北接苏州85公里,处于长江三角洲地带。</p></div></div></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,translateY(100%)表示在垂直方向上从当前位置平移高度的距离;-webkit-transition: all 0.35s ease 表示在 350 毫秒内以 ease 模式执行垂直方向上的平移动作。

此实例的源文件名是 myHtmlB259.html。

261 使用 transition 属性和 translateY()方法实现在图像上推出简介层

此实例主要通过综合使用 transition 属性和 translateY()方法实现当鼠标指针悬浮在图像上时简介层从下向上推出图像的效果。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在第一幅图像上,则该图像对应的简介层从下向上推出该图像至可视范围外,简介层则占据可视窗口的全部区域,如图 261-1 所示;如果鼠标指针悬浮在第二幅图像上,则该图像对应的简介层从下向上推出该图像至可视范围外,简介层则占据可视窗口的全部区域,如图 261-2 所示;如果鼠标指针悬浮在第三幅图像上,则该图像对应的简介层从下向上推出该图像至可视范围外,简介层则占据可视窗口的全部区域。有关此实例的主要代码如下。



图 261-1



图 261-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  *{margin:0;padding:0;}
  body{background-color:black;}
  /*设置盒子样式*/
  .myBox{display:inline-block;padding:0px;border-radius:5px;margin:10px;}
  /*设置简介层样式*/
  [class^='myBrief-']{border-radius:5px;position:relative;display:inline-block;background-
                    color:darkgreen;color:yellow;overflow:hidden;}
  /*设置图像顶部对齐*/
  [class^='myBrief-']>img{vertical-align:top;max-width:100%;}
  /*设置文字样式*/
  p{text-align:left;margin:5px;font-size:14px;}
  h3{margin:5px;}
  [class^='myBrief-'] figcaption{background-color:inherit;
  padding:5px;position:absolute;top:0;bottom:0;left:0;right:0;}
  /*设置动画的参数*/
  [class^='myBrief-'] *{-webkit-transition:all 0.5s ease;}
  /*鼠标指针悬浮时开始执行动画*/
  [class^='myBrief-push-']:hover figcaption{-webkit-transform:translate(0,0);}
  /*向上移动简介*/
  .myBrief-push-up figcaption {-webkit-transform:translateY(100%);}
  /*向上移动图像*/
  .myBrief-push-up:hover>img {-webkit-transform:translateY(-100%);}
 </style></head>
<body><center><div class="myBox"><figure class="myBrief-push-up">
   <img src="img/B259A.jpg"><figcaption><h3>镇远古镇</h3><p>镇远古镇是贵州省黔东南苗族侗族自
治州镇远县名镇,位于舞阳河畔,四周皆山。河水蜿蜒,以S形穿城而过,北岸为旧府城,南岸为旧卫城,远观颇
似太极图。两城池皆为明代所建,现尚存部分城墙和城门。</p></figcaption></figure></div>
 <div class="myBox"><figure class="myBrief-push-up"><img src="img/B259B.jpg">
   <figcaption><h3>丽江古镇</h3><p>丽江古城,又名"大研古镇",世界文化遗产,国家5A级旅游景区,全
国文明风景旅游区示范点。位于中国西南部云南省的丽江市,坐落在丽江坝中部,玉龙雪山下,北倚象山、金虹
山、西枕狮子山。</p></figcaption></figure></div>
 <div class="myBox"><figure class="myBrief-push-up"><img src="img/B259C.jpg">
 <figcaption><h3>西塘古镇</h3><p>西塘是江南六大古镇之一,位于浙江省嘉兴市嘉善县。嘉善位于上海
西南方向与上海零距离接壤,距上海市中心80公里,大虹桥商务区60公里,西至杭州110公里,南濒嘉兴港乍浦
港区35公里,北接苏州85公里,处于长江三角洲地带。</p></figcaption></figure></div></center></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< figcaption >标签是 HTML5 的新标签，它用于定义 figure 元素的标题，通常 figcaption 元素应该被置于 figure 元素的第一个或最后一个子元素的位置；translateY(100%)表示在垂直方向上从当前位置平移高度的距离，此实例即是将简介层全部推入可视窗口内；-webkit-transform：translateY(－100%)表示在垂直方向上从 0 位置平移高度的距离，此实例即是将图像全部推出可视窗口外；-webkit-transition：all 0.5s ease 表示在 500 毫秒内以 ease 模式执行垂直方向上的平移动作，包括图像向外移动和简介层进入可视窗口的动画，此行代码也可写成-webkit-transition：-webkit-transform 0.5s ease。

此实例的源文件名是 myHtmlB260.html。

262 使用 transition 和 opacity 属性实现淡入和淡出的切换效果

此实例主要通过综合使用 transition 属性和 opacity 属性实现当鼠标指针悬浮在图像上时简介层以淡入的效果显示在图像之上。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第一幅图像上，则该图像对应的简介层将以淡入的效果显示在图像之上，同时在其他图像上之前显示的简介层将以淡出的效果消失，如图 262-1 所示；如果鼠标指针悬浮在第二幅图像上，则该图像对应的简介层将以淡入的效果显示在图像之上，同时在其他图像上之前显示的简介层将以淡出的效果消失，如图 262-2 所示；如果鼠标指针悬浮在第三幅图像上，则该图像对应的简介层将以淡入的效果显示在图像之上，同时在其他图像上之前显示的简介层将以淡出的效果消失。有关此实例的主要代码如下。



图 262-1



图 262-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   * { margin: 0; padding: 0; }
  body { background-color: black;}
  /*设置盒子样式*/
  .myBox { display: inline-block; padding: 0px; border-radius: 5px; margin: 10px; }
  /*设置简介层样式*/
  [class^='myBrief-'] { border-radius: 5px; position: relative;display: inline-block;background-
                        color: darkgreen;color: yellow; overflow: hidden; }
  /*设置图像顶部对齐*/
  [class^='myBrief-']> img { vertical-align: top; max-width: 100%;}
  /*设置文字样式*/
  p { text-align: left;margin: 5px; font-size: 14px; }
```

```
    h3 { margin: 5px;}
    [class^='myBrief-'] figcaption{ background-color: inherit;
       padding: 5px; position: absolute; top: 0; bottom: 0;left: 0; right: 0; }
    /*设置动画的参数*/
    [class^='myBrief-'] *{ -webkit-transition: all 5s ease-in-out;}
    /*鼠标指针悬浮时开始执行动画*/
    [class^='myBrief-blur']:hover figcaption{ opacity: 1; }
    /*改变简介层的透明度,opacity从0.0(完全透明)到1.0(完全不透明)*/
    .myBrief-blur figcaption {opacity: 0; }
    /*改变图像的模糊度*/
    .myBrief-blur:hover>img { -webkit-filter: blur(50px);}
   </style></head>
  <body><center><div class="myBox"><figure class="myBrief-blur"><img src="img/B259A.jpg">
  <figcaption><h3>镇远古镇</h3><p>镇远古镇是贵州省黔东南苗族侗族自治州镇远县名镇,位于舞阳河畔,
  四周皆山。河水蜿蜒,以S形穿城而过,北岸为旧府城,南岸为旧卫城,远观颇似太极图。两城池皆为明代所建,
  现尚存部分城墙和城门。</p></figcaption></figure></div>
   <div class="myBox"><figure class="myBrief-blur"><img src="img/B259B.jpg"><figcaption><h3>丽
  江古镇</h3><p>丽江古城,又名"大研古镇",世界文化遗产,国家5A级旅游景区,全国文明风景旅游区示范点。
  位于中国西南部云南省的丽江市,坐落在丽江坝中部,玉龙雪山下,北倚象山、金虹山、西枕狮子山。</p>
  </figcaption></figure></div>
   <div class="myBox"><figure class="myBrief-blur"><img src="img/B259C.jpg"><figcaption><h3>西
  塘古镇</h3><p>西塘是江南六大古镇之一,位于浙江省嘉兴市嘉善县。嘉善位于上海西南方向与上海零距离
  接壤,距上海市中心80公里,大虹桥商务区60公里,西至杭州110公里,南濒嘉兴港乍浦港区35公里,北接苏州
  85公里,处于长江三角洲地带。</p></figcaption></figure></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,opacity：1表示设置简介层为完全不透明；-webkit-filter：blur(50px)表示使用模糊滤镜在淡入淡出时产生模糊特效；-webkit-transition：all 5s ease-in-out表示在5秒内以ease-in-out模式同时执行改变模糊度和透明度的动画。

此实例的源文件名是myHtmlB261.html。

263　使用transition属性和jQuery代码实现折叠和展开多幅图像

此实例主要通过综合使用transition属性和jQuery代码实现类似于售楼手册的折叠和展开多幅图像的特效。当在Google Chrome浏览器中显示该页面时,如果使用鼠标单击蓝色的封面“中国古镇”,将向左展开多幅图像,如图263-1所示；如果在图263-1中使用鼠标单击红色的封面“中国古镇”,将向中间折叠这些图像。有关此实例的主要代码如下。



图　263-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $('.myCover').on('click', function() {
    //收缩或展开大盒子
    $('.myBox').toggleClass('myExpanded');
    //切换封面的两种状态
    $('.myContainer , .myCover').toggleClass('open');
  });});
 </script>
 <style type="text/css">
  *{ margin: 0; padding: 0;}
  body { background-color: lightgray;}
  /*设置大盒子*/
  .myBox { display: block; box-sizing: border-box; background: white;height: 252px;
         width: 150px; padding: 0px; border-radius: 5px; overflow: hidden;
         position: relative; margin: 10px auto;box-shadow: 4px 4px 12px black;
         transition: all 1.3s cubic-bezier(0.53, 0, 0.15, 1.3); }
  /*设置展开后的大盒子宽度*/
  .myExpanded { width:455px; }
  /*设置图像盒子*/
  span{ float: left; height: 250px; width: 150px; margin:1px ; }
  /*设置封面盒子*/
  .myContainer { position: absolute; top: 0px; right:0px; }
  /*设置封面*/
  .myCover{ position: relative; height: 252px; width: 150px; cursor: pointer;
          background-color: blue; border-radius: 5px; }
  /*设置展开后的封面*/
  .myCover.open { background-color: red; }
  img{ border-radius: 5px;}
  /*设置封面标题*/
   p{ position: absolute; font-size: 20px; font-weight: bold; color: white;
     top: 110px;left: 35px; }
 </style></head>
<body><div class="myBox"><div class="myContainer"><div class="myCover"><p>中国古镇</p></div>
</div><span><img src="img/B259A.jpg"></span><span><img src="img/B259B.jpg"></span></div></
body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，float：left 用于使 span 元素向左浮动，如果省略此代码，在 span 元素之间将会出现一定程度的空白；overflow：hidden 用于控制 span 元素在超出大盒子时自动被修剪，并且是不可见的，如果省略此代码，span 元素(图像)将在下面出现，不能表现折叠效果；transition：all 1.3s cubic-bezier(0.53，0，0.15，1.3)用于在使用 jQuery 代码(即单击封面折叠和展开图像时使过渡动画在 1.3 秒内以 cubic-bezier 模式完成大盒子宽度的改变，cubic-bezier 又称三次贝塞尔，主要是为动画生成速度曲线的函数。

此实例的源文件名是 myHtmlB264.html。

264 使用 transition 属性和 target 选择器模拟类似手风琴的折叠特效

此实例主要通过综合使用 transition 属性和 target 选择器实现类似于手风琴的折叠、展开特效。当在 Google Chrome 浏览器中显示该页面时，默认情况下将展开第一幅图像，折叠其他 4 幅图像，如图 264-1 所示；使用鼠标单击纵向标题栏“第二名：伦敦”，则将展开第二幅图像，折叠其他 4 幅图像，

如图 264-2 所示；使用鼠标单击纵向标题栏“第五名：东京”，则将展开第五幅图像，折叠其他 4 幅图像，如图 264-3 所示。当然，使用鼠标单击任意一个纵向标题栏都能够展开对应的图像，折叠其他 4 幅图像。有关此实例的主要代码如下。



图 264-1



图 264-2



图 264-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  * { margin: 0px; padding: 0px;}
  body { background: black; }
  /*设置大盒子样式*/
  #myAccordion { padding: 0; margin: 20px auto; list-style: none; width: 730px; height: 300px;
                 overflow: hidden; position: relative; border: 1px solid lightgreen; }
  /*设置竖栏+图片盒子样式*/
  #myAccordion div { width: 530px; height: 300px; position: relative; }
  /*设置竖栏盒子样式,设置过渡动画时间是1秒*/
  #myAccordion li { width: 50px; height: 300px; float: left;
                    -webkit-transition: 1s; }
  /*设置简介样式(在竖栏盒子里面)*/
  #myAccordion div span { display: block; background: rgba(0, 0, 0, 0.6);
                          width: 460px;height: 45px;position: absolute;bottom: 300px;right: 0;
                          padding: 10px; color: yellow;font-size: 14px;border-top: 1px solid #000; }
  /*设置竖栏标题超链接样式*/
  #myAccordion li div>a { display: block; width: 49px; height: 300px; float: left;
                          position: relative; color: white;text-decoration: none;
                          box-shadow: -10px 0 10px -10px rgba(0, 0, 0, 0.6);}
  /*设置竖栏标题样式*/
  #myAccordion li div>a b { display: block; width: 280px; height: 30px;
                            padding-left: 20px; position: absolute; left: 50px;bottom: 5px;
                            font-size: 20px; font-weight: bold;text-shadow: 0 0 2px rgba(0, 0, 0, 0.6);
                            /*竖栏标题逆时针旋转90°直立起来*/
                            -webkit-transform: rotate(-90deg); -webkit-transform-origin: bottom
                            left; }
  /*设置图像样式*/
  #myAccordion li img { width: 480px; height: 300px; display: block; float: left; }
  /*首次展开p1*/
  #myAccordion li.p1 { width: 530px;}
  /*首次展开p1,显示简介*/
  #myAccordion li.p1 div span { bottom: 0; }
  /*设置p1竖栏超链接背景样式*/
  #myAccordion li a.a1 { background: #460;border-left: 1px solid #682;}
  /*设置p2竖栏超链接背景样式*/
  #myAccordion li a.a2 { background: #793; border-left: 1px solid #9B5; }
  /*设置p3竖栏超链接背景样式*/
  #myAccordion li a.a3 { background: #AC3;border-left: 1px solid #CE5; }
  /*设置p4竖栏超链接背景样式*/
  #myAccordion li a.a4 { background: #CD5;border-left: 1px solid #EF7; }
  /*设置p5竖栏超链接背景样式*/
  #myAccordion li a.a5 { background: #069;border-left: 1px solid #28B; }
  /*设置垂直滑动距离(图像和标题)*/
   #p1:target ~ #myAccordion li, #p2:target ~ #myAccordion li, #p3:target ~ #myAccordion li,
#p4:target ~ #myAccordion li, #p5:target ~ #myAccordion li {
                              width: 50px; }
   #p1:target ~ #myAccordion li.p1, #p2:target ~ #myAccordion li.p2, #p3:target ~ #myAccordion
li.p3, #p4:target ~ #myAccordion li.p4, #p5:target ~ #myAccordion li.p5 { width: 530px; }
   /*设置底部距离(简介)*/
   #p1:target ~ #myAccordion li.p1 div span, #p2:target ~ #myAccordion li.p2 div span, #p3:target ~
#myAccordion li.p3 div span, #p4:target ~ #myAccordion li.p4 div span, #p5:target ~ #myAccordion
li.p5 div span { bottom: 0;}
```

```
</style></head>
<body><div><b id="p1"></b><b id="p2"></b><b id="p3"></b><b id="p4"></b>
  <b id="p5"></b><ul id="myAccordion"><li class="p1">
  <div><a class="a1" href="#p1"><b>第一名：纽约</b></a><img src="img/B266A.jpg"><span>纽约是一座世界级城市，直接影响着全球的经济、金融、媒体、政治、教育、娱乐与时尚界，其中联合国总部也位于该市，因此纽约也被公认为世界之都。</span></div></li>
  <li class="p2"><div><a class="a2" href="#p2"><b>第二名：伦敦</b></a><img src="img/B266B.jpg"><span>伦敦是英国的政治、经济、文化、金融中心和世界著名的旅游胜地，有数量众多的名胜景点与博物馆。伦敦是多元化的大都市，居民来自世界各地，一座种族、宗教与文化的大熔炉城市。</span></div></li>
   <li class="p3"><div><a class="a3" href="#p3"><b>第三名：香港</b></a><img src="img/B266C.jpg"><span>香港是全球高度繁荣的国际大都会之一，全境由香港岛、九龙半岛、新界等3大区域组成。管辖陆地总面积1104.32平方公里，截至2014年末，总人口约726.4万人，人口密度居全球界第三。</span></div></li>
   <li class="p4"><div><a class="a4" href="#p4"><b>第四名：新加坡</b></a><img src="img/B266D.jpg"><span>新加坡是东南亚的一个岛国，被誉为"亚洲四小龙"之一，其经济模式被称为国家资本主义。根据2014年的全球金融中心指数排名报告，新加坡是国际金融中心，也是亚洲重要的服务和航运中心之一。</span></div></li>
   <li class="p5"><div><a class="a5" href="#p5"><b>第五名：东京</b></a><img src="img/B266E.jpg"><span>东京是日本国的政治、经济、文化中心，海陆空交通枢纽，根据建成区面积、人口以及国民生产总值等指标，东京是世界第一大城市，全球最大的经济中心之一。在2014全球城市综合实力排名中位于世界第四。</span></div></li></ul></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transition：1s 用于使过渡动画在1秒内完成每个模块宽度的改变。因为在第一名：纽约中已经通过 href="#p1"指定选择器 #p1：target ～ #myAccordion li、#p1：target ～ #myAccordion li.p1 等中设置的 width 属性值让过渡动画来改变。在 CSS3 中，target 选择器可用于选取当前活动的目标元素，它通常用于响应 URL，后面跟有锚名称 #，指向文档内某个具体的元素，这个被链接的元素就是目标元素（target element）。

此实例的源文件名是 myHtmlB266.html。

265　使用 transition 属性实现图像由彩色变为黑白

此实例主要在 transition 属性中动态改变滤镜-webkit-filter，从而将彩色图像逐渐改变成黑白图像。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针未悬浮在图像上，则显示彩色的图像；如果鼠标指针悬浮在图像上，则彩色图像逐渐变成黑白图像，如图 265-1 所示；反之，如果鼠标指针离开图像，则黑白图像逐渐变成彩色图像。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div { display: inline-block; width: 400px; height: 250px;margin: 5px auto;
       overflow: hidden; background-color: #FFF;border: 1px solid #EEE; position: relative; box-shadow: 5px 5px 5px 1px #999, 0 0 2px rgba(0, 0, 0, 0.06) inset;}
  body { background-color: lightgray;}
  img { -webkit-filter: grayscale(0); -webkit-transition: -webkit-filter 5s ease; }
  img:hover { -webkit-filter: grayscale(1);}
 </style></head>
<body><div><img src="img/B266D.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-filter：grayscale(1)表示将图像完全变成黑白(灰度)效果；-webkit-filter：grayscale(0)表示图像保持原状，没有灰度滤镜效果；-webkit-transition：-webkit-filter 5s ease 表示在 5 秒内以 ease 方式将图像由-webkit-filter：grayscale(0)原始状态改变成-webkit-filter:grayscale(1)黑白状态。

此实例的源文件名是 myHtmlB267.html。

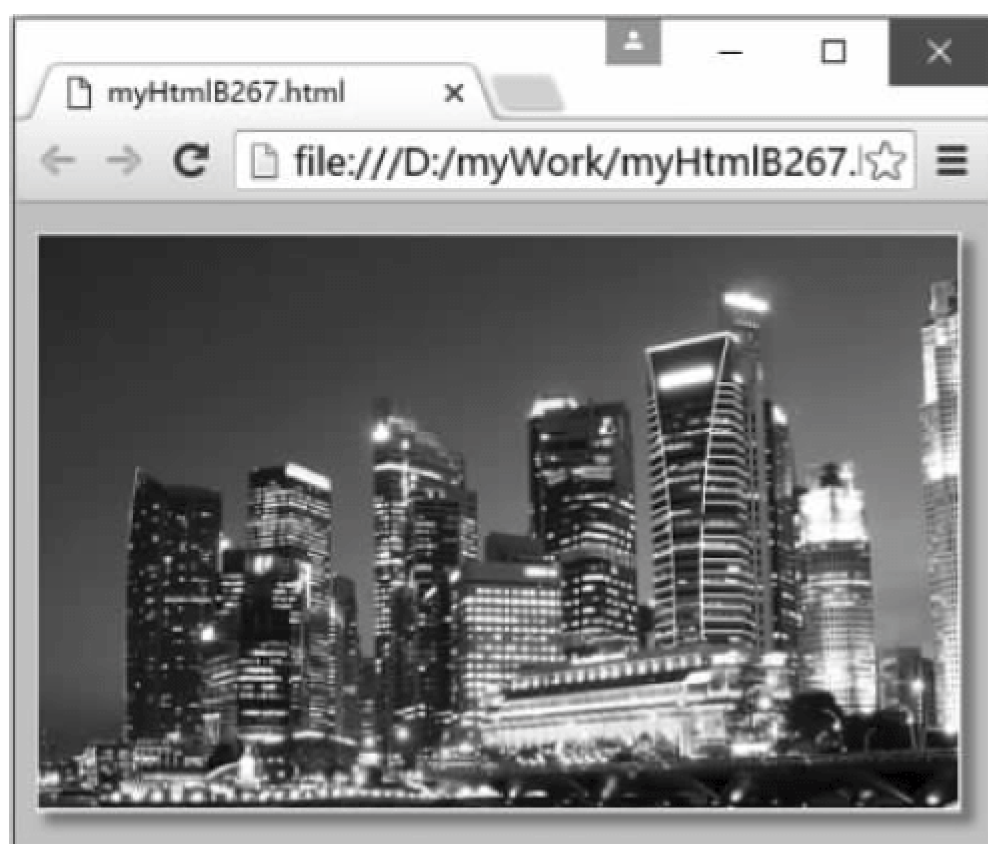


图 265-1

266 使用 transition 属性动态改变圆饼图的百分比

此实例主要在 transition 属性中对圆饼图按照一定的角度进行旋转，从而实现动态改变圆饼图的每部分的百分比(占比)。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指标未悬浮在圆饼上，则显示绿色部分在圆饼中的占比是 87.5%；如果鼠标指标悬浮在圆饼上，则圆饼从 45°顺时针旋转到 135°，因此绿色部分在圆饼中的占比就改变为 62.5%，如图 266-1 所示。有关此实例的主要代码如下。

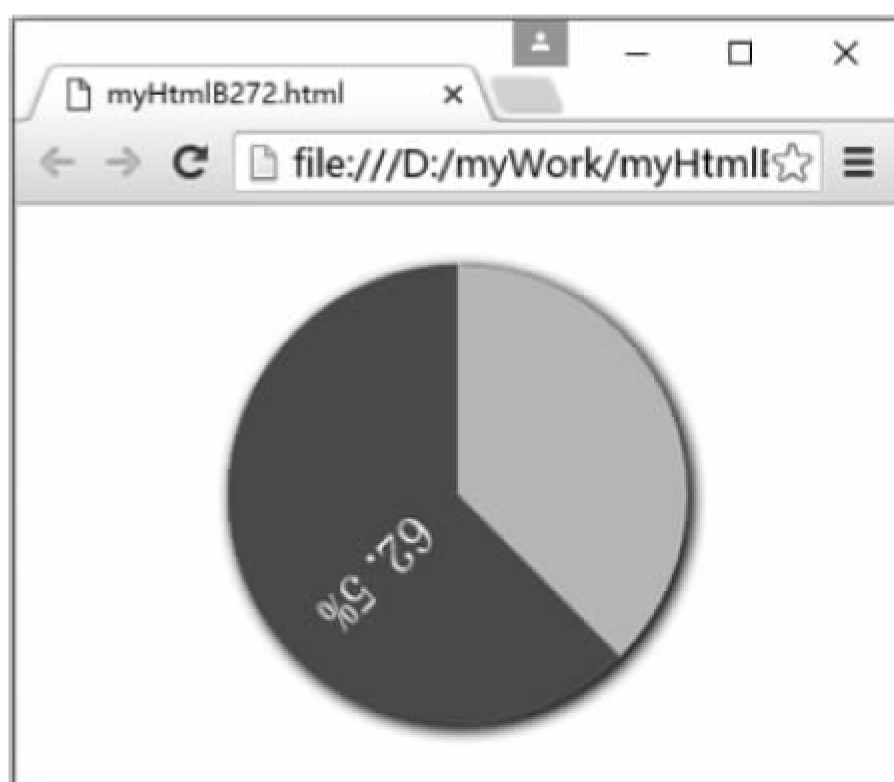


图 266-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
.myPie { position: absolute;top: 50%; left: 50%;transform: translate(-50%, -50%);
        width: 200px; height:200px;line-height: 200px; border-radius: 50%;
        text-align: center; color: white;font-size: 24px; background-color: green;
        overflow: hidden; background-image: linear-gradient(to right, transparent 50%, lightblue
        0); box-shadow: 2px 2px 8px black; }
/*初始旋转45°*/
.myPie:before { content: "87.5%"; position: absolute; top: 0; left: 50%;
              width: 50%; height: 100%;background-color: inherit;
              transform-origin: left; transform: rotate(45deg);}
.myPie:hover:before{ content: "62.5%";transition: transform 1s;
                   transform: rotate(135deg); } /*鼠标指针悬浮时旋转135°*/
</style></head>
<body><div class="myPie"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform：rotate(135deg)表示将元素(圆饼)旋转到135°；transition：transform 1s表示在1秒内以默认方式完成将元素(圆饼)旋转到135°的动作。

此实例的源文件名是myHtmlB272.html。

267　使用transition属性拉伸和收缩文本的下画线

此实例主要在transition属性中改变背景下画线的横坐标位置，从而实现动态拉伸和收缩下画线的效果。当在Google Chrome浏览器中显示该页面时，如果鼠标指针悬浮在《巴山夜雨》、《天云山传奇》上，则将在其底部从左向右拉出红色的下画线，其他下画线则从右向左收缩至消失，如图267-1所示；如果鼠标指针悬浮在其他片名上，也将在其底部从左向右拉出红色的下画线，其他下画线则从右向左收缩至消失。有关此实例的主要代码如下。

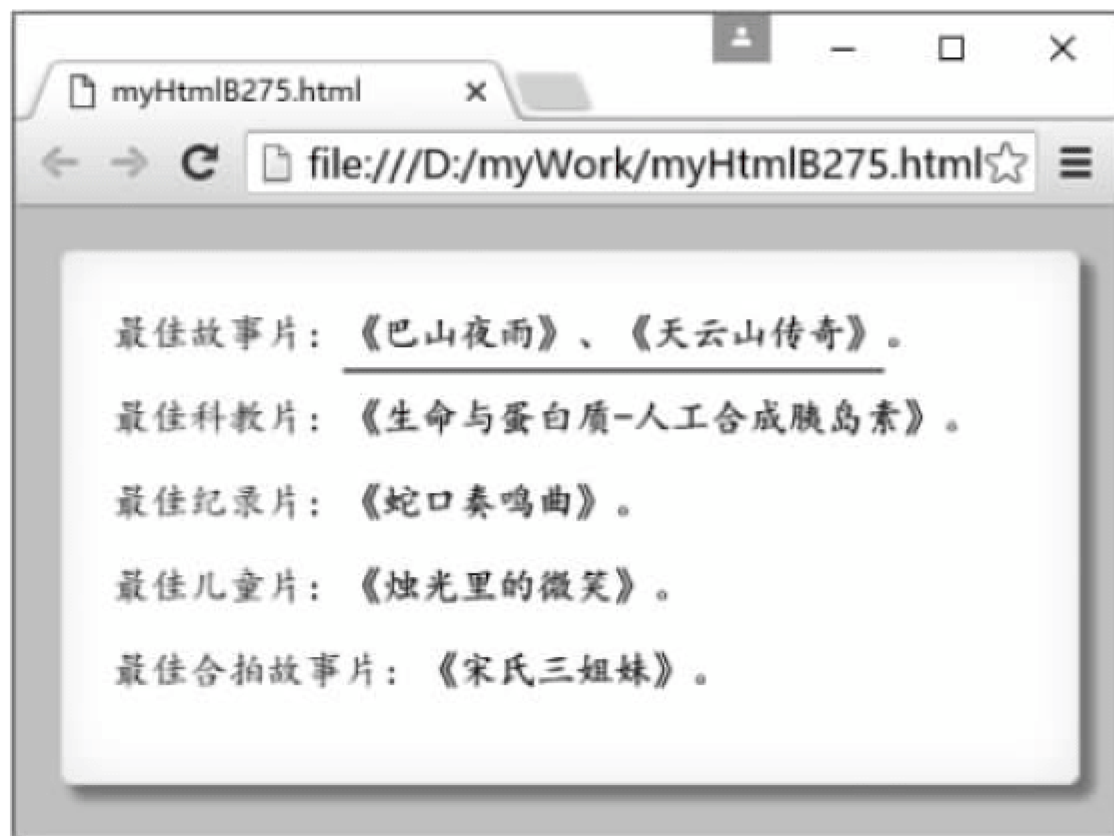


图　267-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /*绘制阴影框*/
  div { display: inline-block; width: 380px; height: 200px;
       padding-left: 20px; padding-right: 20px; padding-bottom: 20px;
```

```
        margin: 10px;background-color: #FFF;border: 1px solid #EEE;
        box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
        position: relative; border-radius: 5px; }
  body { background-color: lightgray; }
  /*取消段落默认的下画线属性,设置字体大小,注意必须与span中的字体大小一致*/
  p {line-height: 35px; text-decoration: none;font-family: 楷体;font-size: 16px; }
  /*定制下画线的风格*/
  span { position: absolute; height: 35px; font-size: 16px;
        background: linear-gradient(red,red) no-repeat;
        background-size: 100% 2px;
        background-position: -30em 2em; transition:1.5s; }
  span:hover { background-position: 0 2em; }
 </style></head>
<body><div><p>最佳故事片:<span>«巴山夜雨»、«天云山传奇»</span>«巴山夜雨»、«天云山传奇»。<br>
        最佳科教片:<span>«生命与蛋白质-人工合成胰岛素»</span>«生命与蛋白质-人工合成胰岛素»。
<br>
        最佳纪录片:<span>«蛇口奏鸣曲»</span>«蛇口奏鸣曲»。<br>
        最佳儿童片:<span>«烛光里的微笑»</span>«烛光里的微笑»。<br>
        最佳合拍故事片:<span>«宋氏三姐妹»</span>«宋氏三姐妹»。<br></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-position:-30em 2em用于隐藏background:linear-gradient(red,red) no-repeat 创建的红色下画线;transition:1.5s 表示在1.5秒内以默认方式将创建的自定义红色下画线移动到background-position:0 2em。此处的transition 属性虽然没有明确指定过渡的属性名称,但它完全等价于 transition:background-position 1.5s。

此实例的源文件名是myHtmlB275.html。

268 使用transition属性实现扇形风格的多级菜单

此实例主要在transition属性中旋转包含子级菜单的无序列表,从而实现滑出扇形风格的子级菜单。当在Google Chrome浏览器中显示该页面时,默认情况下将在左上角显示主菜单,如图268-1所示;如果鼠标指针悬浮在主菜单上,则将逆时针旋转90°,滑出一级菜单,如图268-2所示;如果鼠标指针悬浮在一级菜单上,则将逆时针旋转90°,滑出二级菜单,如图268-3所示;如果鼠标指针悬浮在二级菜单上,如图268-4所示,单击此菜单将弹出一个消息框,响应鼠标单击事件。一旦鼠标指针离开某个菜单,则其子菜单将顺时针旋转90°,恢复初始状态。有关此实例的主要代码如下。

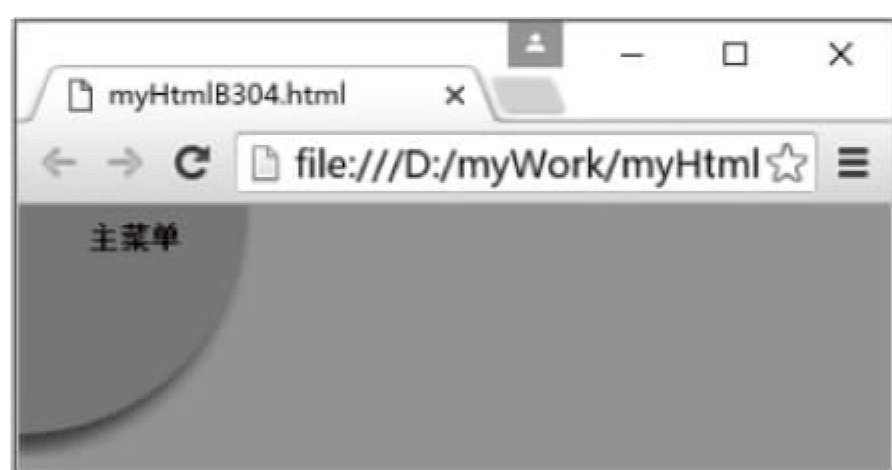


图 268-1



图 268-2

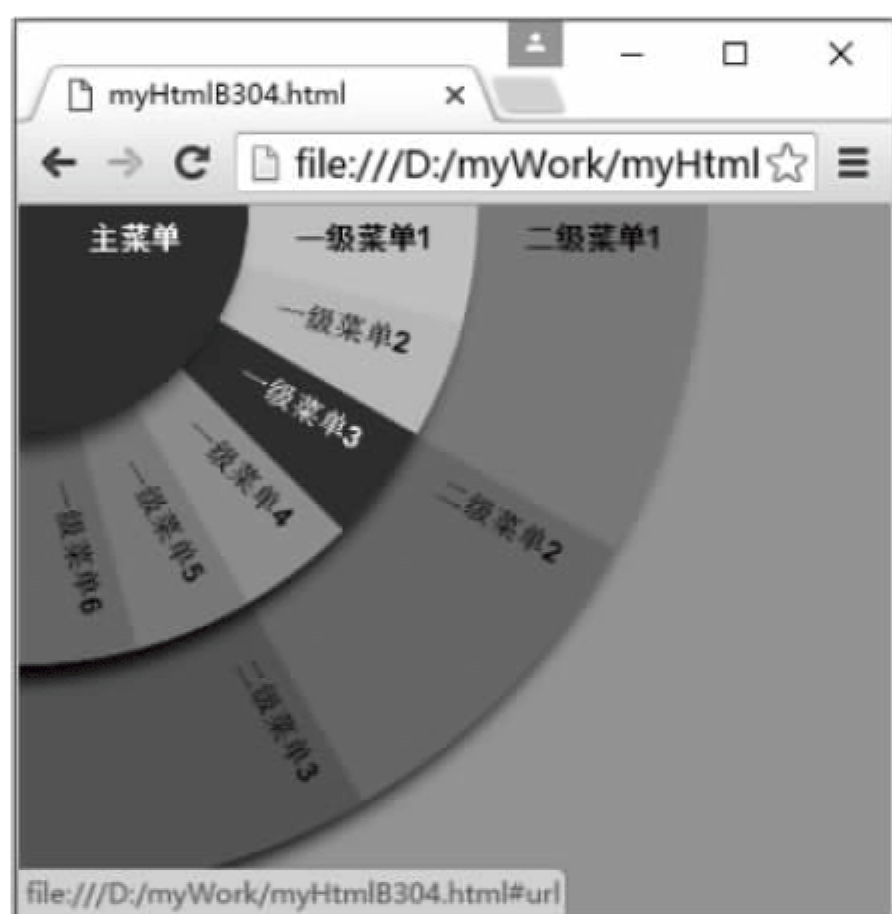


图 268-3



图 268-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
    * { margin: 0px; padding: 0px; }
   body { background: #B1B1B1; margin: 0px;
         padding: 0px; font-size: 14px; color: black; }
   /*设置主菜单盒子*/
   .menuHolder { width: 100px; height: 100px;position: relative; z-index: 100; }
   /*设置主菜单样式*/
   .menuHolder ul { padding: 0; margin: 0; list-style: none; position: absolute;
                    left: 0; top: 0; width: 0; height: 0; }
   /*设置菜单项样式*/
   .menuHolder ul li a { color: #000; text-decoration: none; text-align: center;
                         font: bold 13px/30px arial, sans-serif;
                         box-shadow: -5px 7px 10px rgba(0, 0, 0, 0.5); -webkit-transform-origin: 0
                         0; }
   /*设置子菜单的Z轴顺序,注释此代码即可看到整个扇形结构*/
   .menuHolder ul.p2 { z-index: -1; }
   .menuHolder ul.p3 { z-index: -1; }
   /*设置主菜单样式*/
   .menuHolder li.s1>a { position: absolute; width: 100px;height: 100px;
                          display: block; background: lightseagreen;border-radius: 0 0 100px 0; }
   /*设置一级子菜单样式*/
   .menuHolder li.s2>a { position: absolute; display: block;width: 100px;
                          padding-left: 100px; height: 200px; background: lightgrey;
                          border-radius: 0 0 200px 0; overflow: hidden; }
   /*设置二级子菜单样式*/
   .menuHolder ul.p3 li a { position: absolute; display: block;width: 100px;
                             padding-left: 200px; height: 300px; background: #999;
                             border-radius: 0 0 300px 0; overflow: hidden; }
   /*悬浮时旋转90°滑出子菜单*/
   .menuHolder ul ul { -webkit-transform-origin: 0 0;
                 -webkit-transform: rotate(90deg); -webkit-transition: 1s;}
   /*设置子菜单(项)的旋转角度和背景*/
```

```
.menuHolder li.s2:nth-of-type(6)>a {
     background: #888; -webkit-transform: rotate(75deg); }
.menuHolder li.s2:nth-of-type(5)>a {
     background: #999; -webkit-transform: rotate(60deg);}
.menuHolder li.s2:nth-of-type(4)>a {
     background: #AAA; -webkit-transform: rotate(45deg); }
.menuHolder li.s2:nth-of-type(3)>a {
     background: #BBB; -webkit-transform: rotate(30deg); }
.menuHolder li.s2:nth-of-type(2)>a {
     background: #CCC; -webkit-transform: rotate(15deg); }
.menuHolder .a6 li:nth-of-type(6)>a {
     background: #444; -webkit-transform: rotate(75deg); }
.menuHolder .a6 li:nth-of-type(5)>a {
     background: #555; -webkit-transform: rotate(60deg); }
.menuHolder .a6 li:nth-of-type(4)>a {
     background: #666; -webkit-transform: rotate(45deg); }
.menuHolder .a6 li:nth-of-type(3)>a {
     background: #777; -webkit-transform: rotate(30deg); }
.menuHolder .a6 li:nth-of-type(2)>a {
     background: #888; -webkit-transform: rotate(15deg);}
.menuHolder .a5 li:nth-of-type(5)>a {
     background: #555; -webkit-transform: rotate(72deg); }
.menuHolder .a5 li:nth-of-type(4)>a {
     background: #666; -webkit-transform: rotate(54deg); }
.menuHolder .a5 li:nth-of-type(3)>a {
     background: #777; -webkit-transform: rotate(36deg); }
.menuHolder .a5 li:nth-of-type(2)>a {
     background: #888; -webkit-transform: rotate(18deg); }
.menuHolder .a3 li:nth-of-type(3)>a {
     background: #777; -webkit-transform: rotate(60deg); }
.menuHolder .a3 li:nth-of-type(2)>a {
     background: #888; -webkit-transform: rotate(30deg); }
/* 悬浮时显示一级菜单 */
.menuHolder li.s1:hover ul.p2 { -webkit-transform: rotate(0deg);}
/* 悬浮时显示二级菜单 */
.menuHolder li.s2:hover ul.p3 { -webkit-transform: rotate(0deg);}
/* 设置主菜单悬浮时的背景颜色和文本颜色 */
.menuHolder ul li:hover>a { background: blue; color: #FFF;}
/* 设置一级菜单悬浮时的背景颜色和文本颜色 */
.menuHolder li.s2:hover>a { background: blue; color: #FFF; }
/* 设置二级菜单悬浮时的背景颜色和文本颜色 */
.menuHolder .a6 li:hover>a { background: blue;color: #FFF; }
.menuHolder .a5 li:hover>a { background: blue;color: #FFF; }
.menuHolder .a3 li:hover>a { background: blue;color: #FFF; }
.menuWindow { width: 110px; height: 110px; left: 0; top: 0; z-index: 100;
    overflow: hidden; position: absolute; -webkit-transition: 0s 1s; }
.menuHolder:hover .menuWindow { width: 310px; height: 310px;
                                              -webkit-transition: 0s 0s;}
.p1{ }
.s1{ }
.s2{ }
```

```
</style></head>
<body><div class="menuHolder"><div class="menuWindow"><ul class="p1">
  <li class="s1"><a href="#url">主菜单</a>
   <ul class="p2"><li class="s2">
     <a href="#url"><span>一级菜单 1</span></a>
      <ul class="p3 a6">
       <li><a href="#">二级菜单 1</a></li><li><a href="#">二级菜单 2</a></li>
       <li><a href="#">二级菜单 3</a></li><li><a href="#">二级菜单 4</a></li>
       <li><a href="#">二级菜单 5</a></li><li><a href="#">二级菜单 6</a></li> </ul></li>
    <li class="s2"><a href="#url"><span>一级菜单 2</span></a>
      <ul class="p3 a5">
       <li><a href="#">二级菜单 1</a></li><li><a href="#">二级菜单 2</a></li>
       <li><a href="#">二级菜单 3</a></li><li><a href="#">二级菜单 4</a></li>
       <li><a href="#">二级菜单 5</a></li></ul></li>
    <li class="s2"><a href="#url"><span>一级菜单 3</span></a>
      <ul class="p3 a3">
       <li><a href="#">二级菜单 1</a></li>
         <li><a href="#" onclick="alert('这是二级菜单')">二级菜单 2</a></li>
         <li><a href="#">二级菜单 3</a></li></ul></li>
    <li class="s2">
     <a href="#url"><span>一级菜单 4</span></a>
      <ul class="p3 a3">
       <li><a href="#">二级菜单 1</a></li><li><a href="#">二级菜单 2</a></li>
       <li><a href="#">二级菜单 3</a></li></ul></li>
    <li class="s2">
     <a href="#url"><span>一级菜单 5</span></a>
      <ul class="p3 a3">
       <li><a href="#">二级菜单 1</a></li><li><a href="#">二级菜单 2</a></li>
       <li><a href="#">二级菜单 3</a></li></ul></li>
    <li class="s2">
     <a href="#url"><span>一级菜单 6</span></a>
      <ul class="p3 a3">
       <li><a href="#">二级菜单 1</a></li><li><a href="#">二级菜单 2</a></li>
       <li><a href="#">二级菜单 3</a></li></ul></li></ul></li></ul></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：rotate(90deg)用于旋转90°滑出子级菜单，此处实质上旋转的是整个子级菜单的容器。-webkit-transition：1s表示在1秒内以默认方式执行90°的旋转动作，它相当于-webkit-transition：transform 1s，即以过渡动画方式旋转90°，滑出子级菜单。

此实例的源文件名是myHtmlB304.html。

269　使用 transition 属性实现抽屉风格的滑出菜单

此实例主要在transition属性中改变元素的左内边距，从而实现抽屉风格的滑出菜单。当在Google Chrome浏览器中显示该页面时，如果鼠标指针悬浮在“宾夕法尼亚大学”菜单上，则将增加左内边距，向右滑出菜单，如图269-1所示；如果鼠标指针悬浮在其他大学菜单上，也将增加左内边距，向右滑出菜单。一旦鼠标指针离开某个菜单项，则将恢复菜单项初始状态的左内边距。有关此实例的主要代码如下。



图 269-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  * { margin: 0; padding: 0; }
  body { position: relative; height: 100%;
        background: linear-gradient(to right,cyan, white);}
  /*菜单盒子*/
  .myMenu { list-style: none; display: block;
           position: relative; top: 6px; left: 5px; width:160px;}
  li { margin: 5px 0 0 0; }
  /*设置菜单项样式*/
  .myMenu li a { -webkit-transition: 0.3s ease-out; color: black; display: block; background:
                lightgreen; padding: 7px 15px 7px 10px; width: 120px; font-size: 14px; -webkit-
                border-top-right-radius: 10px; -webkit-box-shadow: 2px 2px 4px #888; -webkit
                -border-bottom-right-radius: 10px;text-decoration: none; }
  /*设置鼠标指针悬浮状态样式*/
  .myMenu li a:hover { background-color: pink;padding: 7px 15px 7px 30px; }
 </style></head>
<body><ul class="myMenu">
 <li><a href="javascript:void(0)">密歇根大学</a></li>
 <li><a href="javascript:void(0)">约翰霍普金斯大学</a></li>
 <li><a href="javascript:void(0)">宾夕法尼亚大学</a></li>
 <li><a href="javascript:void(0)">斯坦福大学</a></li>
 <li><a href="javascript:void(0)">普林斯顿大学</a></li>
 <li><a href="javascript:void(0)">圣安德鲁斯大学</a></li>
 <li><a href="javascript:void(0)">加州理工学院</a></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，padding: 7px 15px 7px 30px 表示鼠标指针悬浮于某个菜单项时，该菜单项的上内边距是 7px、右内边距是 15px、下内边距是 7px、左内边距是 30px。在 CSS3 中，padding 属性用于在一个声明中设置所有内边距属性。下面是其他 3 种内边距的表示形式，padding:10px 5px 15px 表示上内边距是 10px、右内边距和左内边距是 5px、下内边距是 15px；padding:10px 5px 表示上内边距和下内边距是 10px、右内边距和左内边距是 5px；padding:10px 表示上、下、左、右 4 个内边距都是 10px。-webkit-transition: 0.3s ease-out 表示在 300 毫秒内以 ease-out 方式执行改变菜单项左内边距的动作。

此实例的源文件名是 myHtmlB305.html。

270　使用 transition 属性模拟用鼠标操控旋转木马

此实例主要在 transition 属性中根据变动的参数设置 rotateY()和 translateZ()等方法，从而模拟使用鼠标操控木马进行旋转的效果。当在 Google Chrome 浏览器中显示该页面时将以动画的形式展开 3 幅图像，如果按住鼠标左键从右向左移动，则 3 幅图像将沿着椭圆形的轨迹以顺时针方向旋转，如图 270-1 所示；如果按住鼠标左键从左向右移动，则 3 幅图像将沿着椭圆形的轨迹以逆时针方向旋转。有关此实例的主要代码如下。

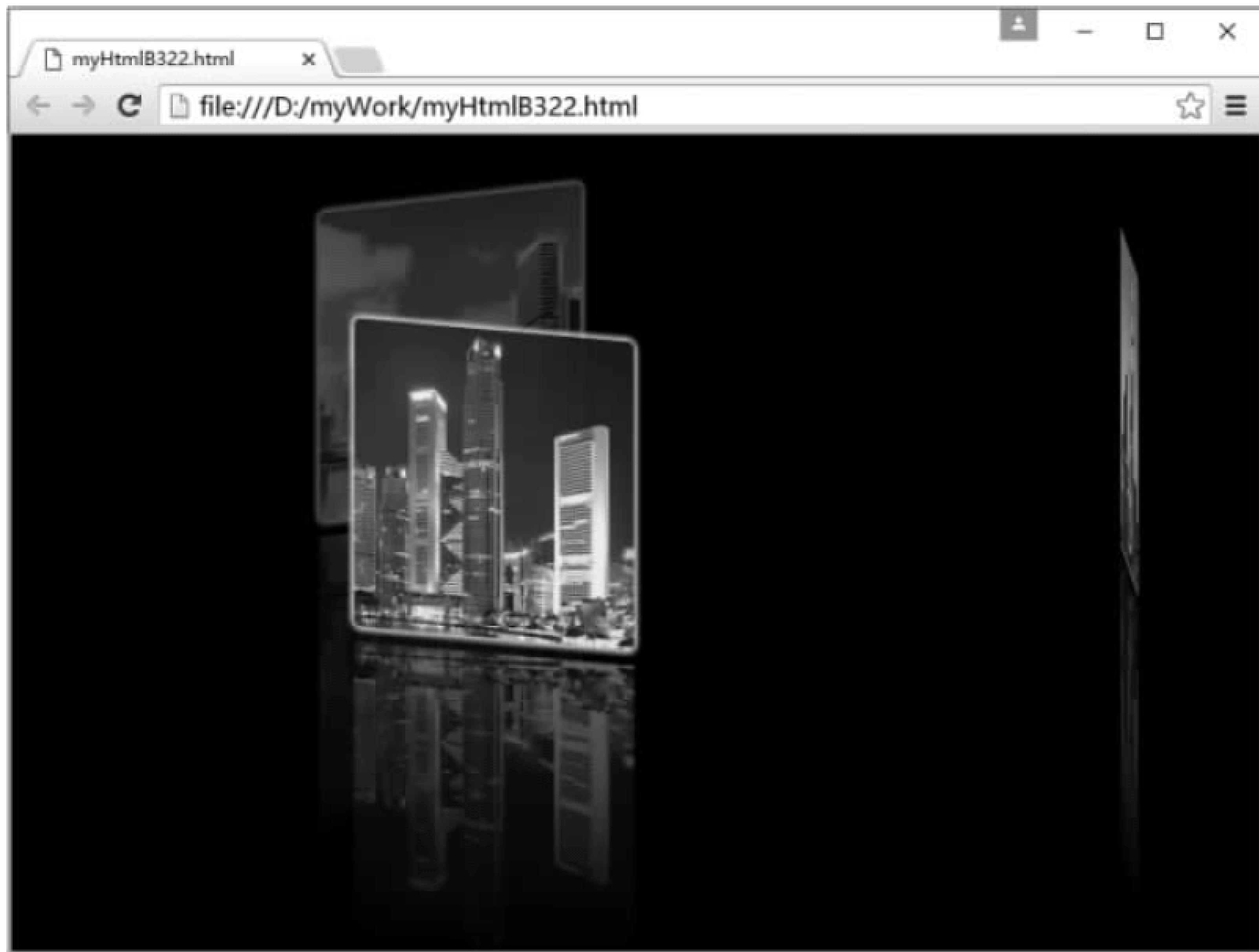


图　270-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script type="text/javascript">
  window.onload = function() {
   var myBox = document.getElementById('myBox');
   var N = 3;
   for (var i = 0; i < N; i++) {
    var myDiv = document.createElement('div');
    myDiv.style.backgroundImage = 'url(img/B322' + (i + 1) + '.jpg)';
    myBox.appendChild(myDiv);
    myDiv.innerHTML = '<span></span>';
    myDiv.children[0].style.backgroundImage = 'url(img/B322' + (i + 1) + '.jpg)';
    (function(myDiv, i) {//实现开始时的 3 幅图像展开动画
     setTimeout(function() {
      myDiv.style.transition = '1s all ease';
      myDiv.style.transform = 'rotateY(' + 360/N * i + 'deg) translateZ(280px)';
     }, 200 * (N - i));
    })(myDiv, i);
   }
   var y = 0;
```

```
        myBox.children[0].addEventListener('transitionend', function() {
          change(y);
        }, false);
        var iSpeedY = 0; var lastY = 0;var timer = null;
        document.onmousedown = function(ev) {//记录鼠标指针位置决定旋转方向
          clearInterval(timer);
          for (var i = 0; i < myBox.children.length; i++) {
            myBox.children[i].style.transition = 'none';
          }
          var disX = ev.clientX - y;
          document.onmousemove = function(ev) {
            y = ev.clientX - disX; change(y/10);
            iSpeedY = y - lastY; lastY = y;
          };
          document.onmouseup = function() {
            document.onmousemove = null;
            document.onmouseup = null;
            timer = setInterval(function() {
              iSpeedY * = 0.95; y += iSpeedY; change(y/10);
            }, 30); };
          return false;
        };
        function change(y) {//平移和旋转图像
          for (var i = 0; i < myBox.children.length; i++) {
            myBox.children[i].style.transform =
                        'rotateY(' + (360/N * i + y) + 'deg) translateZ(280px)';
            var s = Math.abs((360/N * i + y) % 360);
            s > 180 && (s = 360 - s);
            var scale = (180 - s)/180 * 0.8 + 0.2;
            myBox.children[i].style.opacity = scale; } } };
      </script>
      <style type = "text/css">
        * { margin: 0; padding: 0; list - style: none; }
       body { background: #000; }
        #myBox { width: 200px; height: 200px; position: absolute; margin - left: - 65px;
                 top: 80px; left: 50 % ;background: transparent;
                 transform: rotateY(0deg) rotateX( - 10deg); transform - style: preserve - 3d; }
        #myBox div { position: absolute;left: 0; top: 0;width: 100 % ; height: 100 % ;
                     box - shadow: 0 0 5px 2px #FFF; border - radius: 5px; }
       /* 产生倒影效果 */
        #myBox div span { position: absolute;left: 0; top: 0; width: 100 % ; height: 100 % ;
                          transform: translateY(10px) scale(1, - 1);transform - origin: center bottom;
                          - webkit - mask: - webkit - linear - gradient(rgba(0, 0, 0, 0), rgba(0, 0, 0, 1));
                          opacity: 0.4; }
    </style></head>
    <body><div id = "myBox"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，rotateX(angle)表示沿着 X 轴按照指定的角度进行 3D 旋转；rotateY(angle)表示沿着 Y 轴按照指定的角度进行 3D 旋转；translateZ(z)表示根据参数值进行 Z 轴 3D 转换；transform-style：preserve-3d 规定在 3D 空间中子元素将保留其 3D 位置；myDiv.style.transition='1s all ease'表示在 1 秒内以 ease 方式根据变动的参数执行 rotateX()、rotateY()、translateZ()等动作。

此实例的源文件名是 myHtmlB322.html。

271　使用 transition 属性实现响应 Metro 风格的模块

此实例主要在过渡动画 transition 中调用 rotateY()、scale()等方法，从而实现在 Metro 风格的布局中以动态效果显示和关闭模块对应的窗口。当在 Google Chrome 浏览器中显示该页面时，Metro 风格的模块布局如图 271-1 所示，使用鼠标单击任一模块，例如"电子邮件"，则将以旋转放大的方式滑出"电子邮件"对应的窗口，如图 271-2 所示；单击该窗口右上角的关闭按钮，则将关闭"电子邮件"对应的窗口，返回到 Metro 风格的主界面，如图 271-1 所示。使用鼠标单击其他模块将实现类似的功能。有关此实例的主要代码如下。



图　271-1



图　271-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
```

```
$ (document).ready(function() {
 var myBox = $ ('.myBox');
  $ ('.myMetro li').each(function() {
   var myColor = $ (this).css('backgroundColor');
   var myContent = $ (this).html();
   $ (this).click(function() {//响应单击打开模块
    myBox.css('backgroundColor', myColor);
    myBox.addClass('open');
    myBox.find('p').html(myContent);
   });
   $ ('.close').click(function() {//响应单击关闭模块
    myBox.removeClass('open');
    myBox.css('backgroundColor', 'transparent');
});});});
</script>
<style type = "text/css">
 body { background: #363B48; }
 ul, ul li { list-style: none; }
 .myMetro { width: 630px; margin-top: 50px; }
 /*设置模块的基本样式*/
 .myMetro li { cursor: default; position: relative; text-align: center;
             margin: 0 10px 10px 0; width: 150px; height: 150px;
             color: #FFFFFF; float: left; cursor: pointer; }
 /*设置模块中的文本样式*/
 .myMetro li span { color: rgba(255, 255, 255, 0.8); font-size: 30px;
                  line-height: 150px; text-align: center;}
 /*设置第 1 个模块样式*/
 .myMetro li:first-child { background: #00B6DE; }
 /*设置第 2 个模块样式*/
 .myMetro li:nth-child(2) { background: #56DEA7; width: 310px; }
 /*设置第 3 个模块样式*/
 .myMetro li:nth-child(3) { background: #FF7659; margin: 0; }
 /*设置第 4 个模块样式*/
 .myMetro li:nth-child(4) { background: #F8CD36; }
 /*设置第 5 个模块样式*/
 .myMetro li:nth-child(5) { background: #F26175; }
 /*设置第 6 个模块样式*/
 .myMetro li:nth-child(6) { background: #5CA7DF; }
 /*设置第 7 个模块样式*/
 .myMetro li:last-child { background: #9E7AC2; margin: 0; }
 /*弹出窗口的基本设置*/
 .myBox { display: table; top: 0; visibility: hidden; top: 0; left: 0;
        z-index: -1; transform: rotateY(180deg) scale(0.1); position: absolute;
        width: 100%; height: 100%;opacity: 0; transition: 1s all; }
 /*弹出窗口中的文本设置*/
 .myBox p {display: table-cell; vertical-align: middle;
         font-size: 64px; color: #FFFFFF;text-align: center; margin: 0;
         opacity: 0; transition: 0.2s; transition-delay: 0.2s; }
 /*弹出窗口的关闭按钮设置*/
 .myBox .close { display: block; cursor: pointer; position: absolute;
              border: 3px solid rgba(255, 255, 255, 1); border-radius: 50%;
              top: 50px; right: 50px; width: 50px; height: 50px;transform: rotate(45deg);
              transition: 0.2s; transition-delay: 0.2s; opacity: 0; }
```

```
    /*绘制弹出窗口的关闭按钮的左斜线*/
    .myBox .close::before { content: ""; display: block; position: absolute;
                            background-color: rgba(255, 255, 255, 1);
                            width: 80%; height: 6%;left: 10%; top: 47%; }
    /*绘制弹出窗口的关闭按钮的右斜线*/
    .myBox .close::after { content: ""; display: block; position: absolute;
                           background-color: rgba(255, 255, 255, 1);
                           width: 6%; height: 80%; left: 47%; top: 10%; }
    /*显示模块的弹出窗口*/
    .myBox.open { left: 0; top: 0; visibility: visible; opacity: 1;z-index: 999;
                  transform: rotateY(0deg) scale(1); width: 100%; height: 100%; }
    .myBox.open .close, .myBox.open p { opacity: 1; }
   </style></head>
  <body><ul class="myMetro">
      <li><span>游戏专区</span></li><li><span>个人设置</span></li>
      <li><span>电子邮件</span></li><li><span>消息提醒</span></li>
      <li><span>音乐专区</span></li><li><span>个人收藏</span></li>
      <li><span>个人照片</span></li></ul>
  <div class="myBox"><span class="close"></span><p></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，rotateY(180deg)表示沿着 Y 轴旋转 180°；scale(0.1)表示按照 0.1 比例缩放窗口；transition：1s all 表示在 1 秒完成动画转换，即将 transform：rotateY(180deg) scale(0.1)过渡到 transform：rotateY(0deg) scale(1)。

此实例的源文件名是 myHtmlB334.html。

272　使用 transition 属性实现菜单栏的折叠和展开

此实例主要在 transition 属性中改变菜单栏的宽度等属性值，从而实现菜单栏的折叠和展开。当在 Google Chrome 浏览器中显示该页面时，单击展开按钮“＋”，如图 272-1 所示，则将展开菜单栏的所有菜单项，如图 272-2 所示；单击菜单栏左端的关闭按钮“×”，如图 272-2 所示，则将折叠菜单栏的所有菜单项，如图 272-1 所示。有关此案例的主要代码如下。

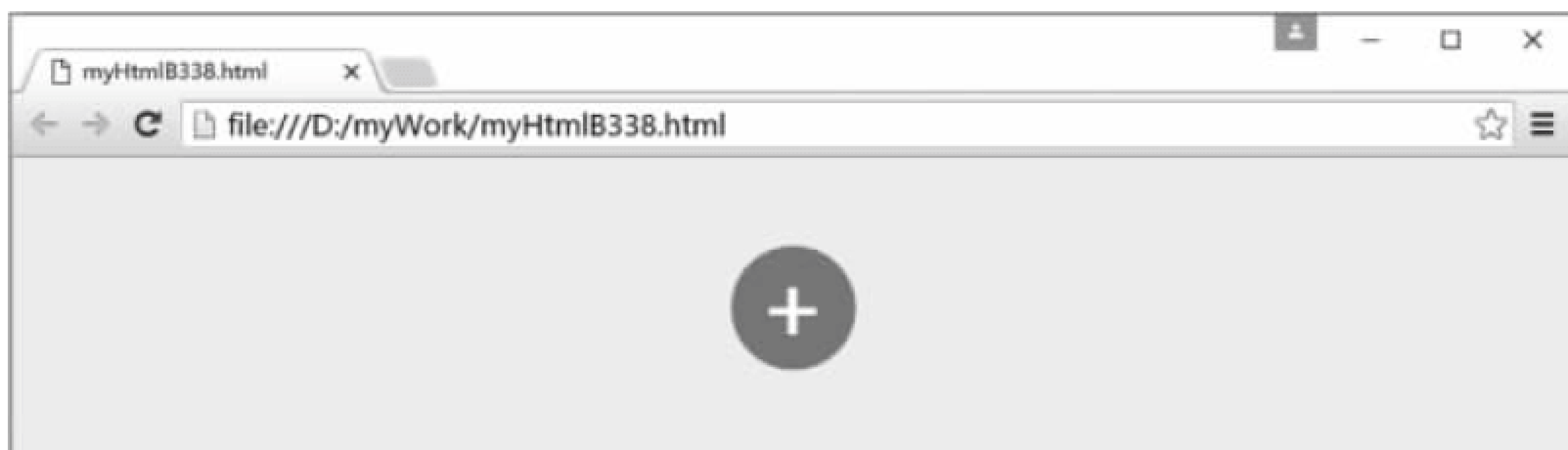


图　272-1

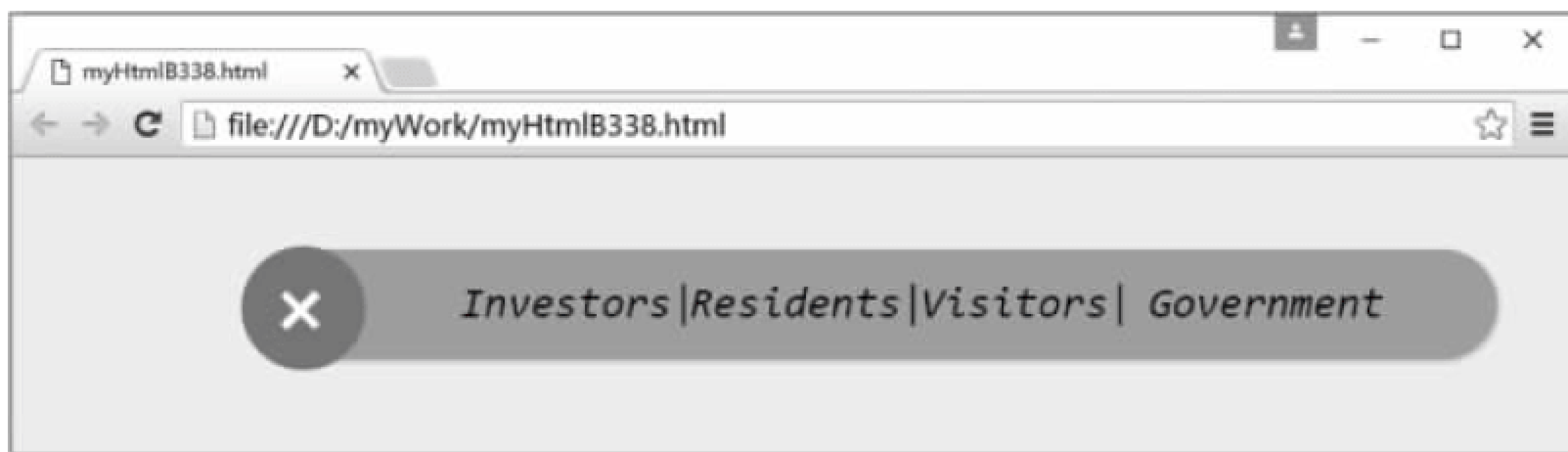


图　272-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $(".myButton").click(function() {
    $(this).toggleClass("active");
    $(".myMenus").toggleClass("open");
  });});
 </script>
 <style type="text/css">
  body { background-color: #F2F3F4; }
  /*设置按钮菜单盒子的基本样式*/
  #myBox { text-align: center;font-family: 'Consolas';
          margin: 50px auto; width: 100%;
          max-width: 670px; min-width: 550px;}
  /*设置按钮的基本样式*/
  .myButton { width: 70px; height: 70px; border-radius: 50%;
             text-align: center; background-color: #3AB09E;
             color: #FFFFFF; font-size: 3.5em; position: relative;
             left: 50%; margin-left: -35px; z-index: 1; }
  .myButton, .myMenus { transition: all 1s cubic-bezier(0.87, -0.41, 0.19, 1.44);}
  /*设置展开按钮*/
  .myButton:after { content: "+"; }
  /*设置关闭按钮*/
  .myButton.active { transform: rotate(45deg); left: 60px; }
  /*设置菜单栏*/
  .myMenus {width: 0%; overflow: hidden;height: 36px; list-style: none;
  padding: 16px 10px 10px 50px; background-color: cyan;
  box-shadow: 1px 1px 1px 1px #DCDCDC; margin: -68px 0 0 45%; border-radius: 2em; }
  /*显示菜单栏*/
  .myMenus.open { width: 95%; margin: -68px 0 0 5%;overflow: hidden; }
  /*隐藏菜单项*/
  .myMenus li { display: none; }
  /*显示菜单项*/
  .myMenus.open li { width: 120px; font-size: 24px;display: inline-block; }
 </style></head>
<body><div id="myBox"><div class="myButton"></div>
 <ul class="myMenus"><li><i>Investors</i>|</li><li><i>Residents</i>|</li>
  <li><i>Visitors</i>|</li><li><i>Government</i></li></ul></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(this).toggleClass("active")中的toggleClass()方法用于在菜单栏的关闭按钮和打开按钮之间进行切换。toggleClass()方法是jQuery的一个方法，它用于对设置或移除被选元素的一个或多个类进行切换，该方法检查每个元素中指定的类，如果不存在则添加类，如果已设置则删除，这就是所谓的切换效果。transition: all 1s cubic-bezier(0.87, −0.41, 0.19, 1.44)表示在1秒内以cubic-bezier(0.87, −0.41, 0.19, 1.44)方式执行改变菜单栏宽度等属性值的所有动作。

此实例的源文件名是myHtmlB338.html。

273　使用 transition 属性模仿光柱划过夜空的效果

此实例主要通过在 transition 中改变渐变图形(光柱)的位置模仿光柱划过夜空的效果。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在图像上,则渐变图形(光柱)将从左上角划向右下角,直到消失,如图 273-1 所示;如果鼠标指针离开图像,则渐变图形(光柱)将从右下角划向左上角,直到消失。有关此实例的主要代码如下。

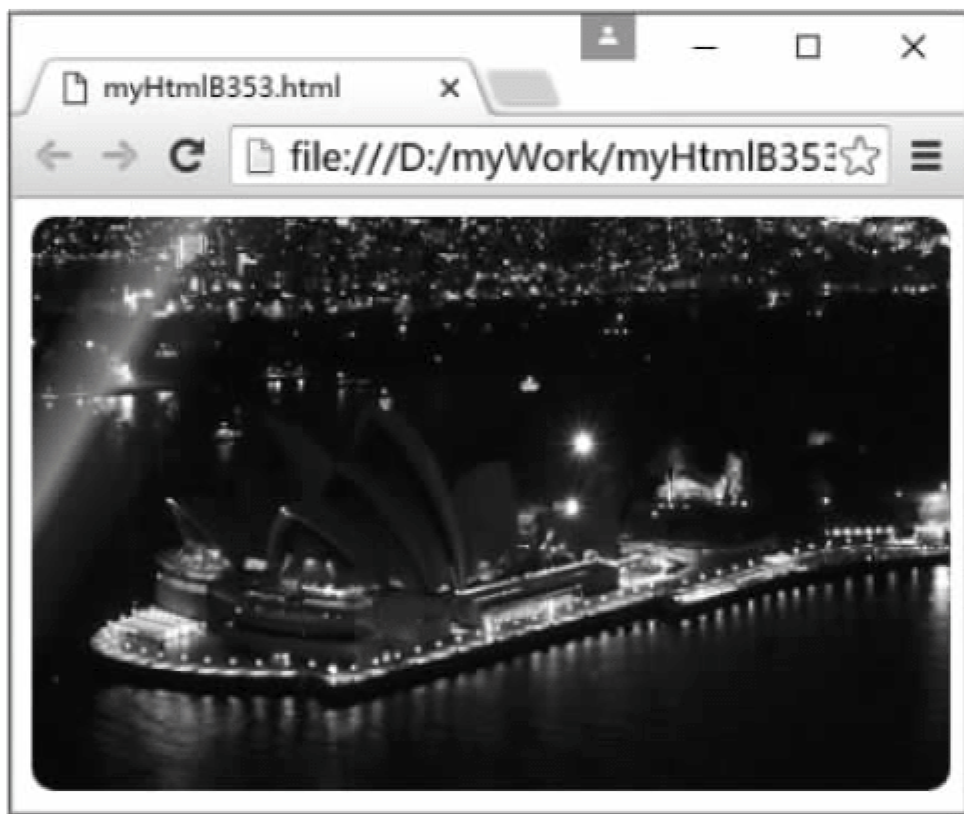


图　273-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*设置背景包含渐变图形(光柱)和图像的div块的基本样式*/
  .myAurora { width: 400px; height: 250px;border-radius: 10px;
            background: -webkit-linear-gradient(-30deg, rgba(255, 255, 255, 0) 0, rgba(255, 255,
            255, 0) 230px, rgba(255, 255, 255, 0.45) 260px, rgba(255, 255, 255, 0.1) 270px, rgba(255,
            255, 255, 0) 300px, rgba(255, 255, 255, 0) 310px) no-repeat, url(img/B353.jpg) no-
            repeat; background-size: 100% 100%;transition:2s background-position ease-in-out;
            background-position: -300px 0,0 0; }
  /*当鼠标指针悬浮时改变渐变图形的位置*/
  .myAurora:hover { background-position:300px 0,0 0; }
 </style></head>
<body><div align="center">
    <div class="myAurora" align="center"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-linear-gradient (−30deg, rgba(255, 255, 255, 0) 0, rgba(255, 255, 255, 0) 230px, rgba(255, 255, 255, 0.45) 260px, rgba (255, 255, 255, 0.1) 270px, rgba(255, 255, 255, 0) 300px, rgba(255, 255, 255, 0) 310px)用于创建一个倾斜 30°的线性渐变图形(光柱),0.45 是透明度值(范围为 0~1,如果希望光柱强,此值可以大一些,反之则小一点),−30deg 表示渐变图形的渐变方向,如果将此值设置为 30deg,即 background: -webkit-linear- gradient(30deg, rgba(255, 255, 255, 0) 0, rgba(255, 255, 255, 0) 230px, rgba(255, 255, 255, 0.45) 260px, rgba(255, 255, 255, 0.1) 270px, rgba(255, 255, 255, 0) 300px, rgba(255, 255, 255, 0) 310px) no-repeat, url(img/B353.jpg) no-repeat,则当鼠标指针悬浮在图像上时渐变图形(光柱)将从左下角划向右上角,直到消失;当鼠标指针离开图像时渐变图形(光柱)将从右上角划向左下角,直到消失。background-position:−300px 0,0 0 用于设置渐变图形和

图像的位置，逗号之前的坐标用于设置渐变图形（光柱）的位置，逗号之后的坐标用于设置图像的位置。transition:2s background-position ease-in-out 表示在两秒内以 ease-in-out 方式完成 background-position：-300px 0,0 0 到 background-position：300px 0,0 0 的改变。

此实例的源文件名是 myHtmlB353.html。

274 使用 transition 属性逐字旋转切换文本的状态

此实例主要在 transition 中执行改变字符角度和颜色的动作，从而实现在旋转的过程中逐字将绿色的文字变成红色的文字。当在 Google Chrome 浏览器中显示该页面时，"CSS3 炫酷实例集锦"字符串将以绿色显示，如果鼠标指针悬浮在该字符串上，则将从左向右把每个字符旋转 360°，并将其颜色改变为红色，如图 274-1 所示；当鼠标指针离开该字符串时执行相反的动作。有关此实例的主要代码如下。



图 274-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   var myTitle=$("a").text(),myChar="";
   //获取每个字符(10 为字符串的字符个数)
   for(var i=0;i<10;i++){ myChar +="<span>"+myTitle.substr(i,1)+"</span>"; }
    $("a").html(myChar);
    for(var i=1;i<=$(".myRotate span").length;i++){
      //设置每个字符需要等待的时间(注意与-webkit-transition: 1s 适配)
      $(".myRotate span:nth-child("+i+")").css("-webkit- transition-delay",((i-1)*0.2)+"s");
  }});
  </script>
  <style type="text/css">
   /* 设置字符串的基本样式 */
   .myRotate{ margin-top:50px;outline: none; position: relative; font-weight: bold; display: inline-block;text-decoration: none; text-transform: uppercase; font-size: 48px;text-shadow: 2px 2px 10px black;}
   /* 设置字符的基本样式 */
   .myRotate span{ display: inline-block; position: relative;
      -webkit-transform:rotateY(0deg); -webkit-transition: 1s; color: darkgreen;}
   /* 设置鼠标指针悬浮在字符串上时字符的基本样式 */
   .myRotate:hover span { color: red; -webkit-transform:rotateY(360deg); }
 </style></head>
<body><div align="center"><a class="myRotate" href="#">CSS3 炫酷实例集锦</a></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：rotateY(360deg)用于将每个字符旋转 360°；-webkit-transition：1s 表示在 1 秒内完成改变字符颜色和旋转的动作；$(".myRotate span:nth-child("+i+")").css("-webkit- transition-delay",((i-1)*0.2)+"s")中的 transition-delay 属性用于规定在过渡动画开始之前需要等待的时间，以秒或毫秒计。

此实例的源文件名是 myHtmlB354.html。

275　使用 transition 属性放大显示当前菜单项

此实例主要在 transition 中执行 scaleX()和 scaleY()方法的放大动作，从而实现在鼠标指针悬浮时以动画的效果放大当前菜单项。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第二个菜单项上，则放大效果如图 275-1 所示；如果鼠标指针悬浮在其他菜单项上，也将实现类似的功能。有关此实例的主要代码如下。



图　275-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  *, *::before, *::after { box-sizing: border-box; }
  *{margin:0;padding: 0;}
  ol,ol li{list-style: none;}
  /*设置盒子的基本样式*/
  .myMenu { margin:10px auto; width: 385px; height: 308px;border-radius: 10px;
           background: -webkit-linear-gradient(top, #3A404D, #181C26);
           box-shadow: 0 7px 30px rgba(62, 9, 11, 0.3); }
  /*设置标题的基本样式*/
  .myMenu h1 { font-size: 20px; color: yellow; padding: 12px 13px 18px; }
  .myMenu ol { /*设置计数器值,默认为 0*/
               counter-reset: myMenu; }
  /*设置每行的基本样式*/
  .myMenu ol li { position: relative; z-index: 1; font-size: 14px;
                  /*自增计数*/
                  counter-increment: myMenu; padding: 18px 10px 18px 50px; }
```

```
  /*设置序号的基本样式*/
  .myMenu ol li::before { content: counter(myMenu); position: absolute;
                          z-index: 2; top: 15px; left: 15px;width: 20px; height: 20px;line-height: 20px;
                          color: #C24448; background: #FFF; border-radius: 20px;text-align: center; }
  /*设置书名的基本样式,mark是HTML5的固有标签*/
  .myMenu ol li mark { position: absolute; z-index: 2; width: 320px; height: 48px;
                       top: 0; left: 0; margin: 0; padding: 18px 10px 18px 50px;
                       /*注意此处不能设置背景颜色*/
                       background: none; color: #FFF; overflow: hidden;
                       text-overflow:ellipsis; white-space:nowrap; }
  /*设置单价的基本样式,small是HTML5的固有标签*/
  .myMenu ol li small { position: relative; color:#FFF;
                        z-index: 2; display: block; text-align: right; }
  /*设置鼠标指针悬浮前每行的基本样式*/
  .myMenu ol li::after {content: ''; position: absolute; top: 0;left: 0;
                        width: 100%;height: 100%;box-shadow: 0 3px 0 rgba(0, 0, 0, 0.08);
                        -webkit-transition:all 0.3s ease-in-out; }
  /*设置鼠标指针悬浮前第1行的背景颜色*/
  .myMenu ol li:nth-child(1),.myMenu ol li:nth-child(1)::after{
           background: #FA6855; }
  /*设置鼠标指针悬浮前第2行的背景颜色*/
  .myMenu ol li:nth-child(2),.myMenu ol li:nth-child(2)::after {
           background: #E0574F; }
  /*设置鼠标指针悬浮前第3行的背景颜色*/
  .myMenu ol li:nth-child(3),.myMenu ol li:nth-child(3)::after{
           background: #D7514D; }
  /*设置鼠标指针悬浮前第4行的背景颜色*/
  .myMenu ol li:nth-child(4),.myMenu ol li:nth-child(4)::after{
           background: #CD4B4B; }
  /*设置鼠标指针悬浮前第5行的背景颜色*/
  .myMenu ol li:nth-child(5),.myMenu ol li:nth-child(5)::after{
           background: #C24448; border-radius: 0 0 10px 10px; }
  /*设置当前行位于最顶层*/
  .myMenu ol li:hover { z-index: 2; overflow: visible; }
  /*设置鼠标指针悬浮每行后的样式*/
  .myMenu ol li:hover::after { background-color:green; border-radius: 5px;
                               -webkit-transform: scaleX(1.06) scaleY(1.03);}
 </style></head>
<body><div class="myMenu" align="center">
 <h1>静心阅读5月份畅销图书排行榜</h1>
 <ol><li><mark>HTML5+CSS3从入门到精通</mark><small>¥47.50</small></li>
     <li><mark>HTML5与CSS3实战指南</mark><small>¥35.55</small></li>
     <li><mark>深入浅出HTML5编程</mark><small>¥82.80</small></li>
     <li><mark>HTML5移动Web开发指南</mark><small>¥50.20</small></li>
     <li><mark>从零开始学HTML5+CSS3</mark><small>¥58.30</small></li></ol></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，scaleX(1.06)表示在X轴方向将元素(菜单项盒子)放大1.06倍；scaleY(1.03)表示在Y轴方向将元素(菜单项盒子)放大1.03倍；-webkit-transition: all 0.3s ease-in-out表示在300毫秒内以ease-in-out模式完成改变菜单项大小、颜色及其他属性的动作。

此实例的源文件名是myHtmlB355.html。

276　使用 transition 属性显示或隐藏侧边栏菜单项

此实例主要在 transition 中执行 scale()方法的放大动作，从而实现隐藏或显示侧边栏菜单项的效果。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标单击右上角的菜单按钮，则将从右上角滑出侧边栏菜单，如图 276-1 所示；如果使用鼠标单击侧边栏右上角的关闭按钮，如图 276-1 所示，则侧边栏将收缩成菜单按钮。有关此实例的主要代码如下。

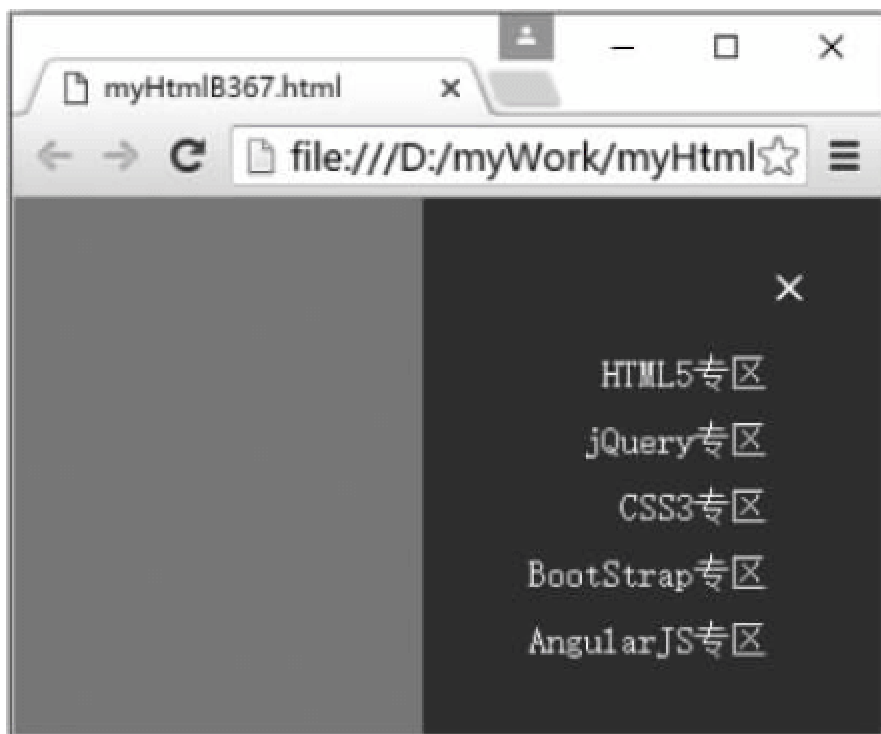


图　276-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $('.myMenu').click(function() { /*响应鼠标单击小圆形菜单按钮*/
    $('.myMenu').toggleClass('myMenu_clicked');
    $('.myLine1').toggleClass('myLine1_clicked');
    $('.myLine2').toggleClass('myLine2_clicked');
    $('.myLine3').toggleClass('myLine3_clicked');
    $('.myMenu_list').toggleClass('myMenu_list_clicked');
  });})
 </script>
 <style type="text/css">
  body { background-color: lightseagreen; }
  /*设置盒子的基本样式*/
  .myBox { background-color: transparent; position: fixed;
          width: 200px; height: 100%; top: 0px; right: 0px;
          /*小圆形盒子放大后自动裁剪超出部分*/
          overflow: hidden !important; z-index: 5; }
  /*设置小圆形盒子的基本样式*/
  .myMenu { float: right; margin: 15px 15px 0 0; height: 49px; width: 49px;
           border-radius: 50%; background-color: blue; border: none; cursor: pointer;
           /*隐藏时的过渡动画*/
           transition: all 0.5s ease-out; }
  /*单击小圆形盒子显示侧边栏*/
  .myMenu_clicked { transform: scale(200);
                   /*显示时的过渡动画*/
                   transition: all 0.35s ease-in; cursor: default; }
  /*绘制第 1 条横线*/
```

```
    .myLine1{ background: #FFF; height: 2px; width: 15px; position: absolute;
             right: 33px;top: 33px; transition: all 0.3s; }
    /*绘制第 2 条横线*/
    .myLine2 {background: #FFF; height: 2px; width: 15px;
             position: absolute; opacity: 1; right: 33px; top: 38px;
             transition: opacity 0.5s; }
    /*绘制第 3 条横线*/
    .myLine3 {background: #FFF; height: 2px; width: 15px;
             position: absolute; right: 33px; top: 43px;}
    /*单击之后第 1 条横线旋转 45°交叉*/
    .myLine1_clicked { animation: closetop 1s forwards alternate; cursor: pointer; }
    /*单击之后隐藏第 2 条横线*/
    .myLine2_clicked { opacity: 0; cursor: pointer; transition: opacity 0.5s; }
    /*单击之后第 3 条横线旋转 45°交叉*/
    .myLine3_clicked { animation: closebottom 1s forwards alternate;
                      cursor: pointer; }
    /*第 1 条横线旋转 45°的关键帧*/
    @keyframes closetop {0% { transform: translateY(5px) rotate(0deg); }
                        25% { transform: translateY(5px) rotate(0deg); }
                        75% { transform: translateY(5px) rotate(-45deg); }
                        100% {transform: translateY(5px) rotate(-45deg); } }
    /*第 3 条横线旋转 45°的关键帧*/
    @keyframes closebottom { 0% { transform: translateY(0px) rotate(0deg); }
                           25% { transform: translateY(-5px) rotate(0deg); }
                           75% { transform: translateY(-5px) rotate(45deg); }
                           100% { transform: translateY(-5px) rotate(45deg); } }
    /*设置侧边栏菜单项盒子的基本样式*/
    .myMenu_list { visibility: hidden; position: absolute; right: 50px; top: 40px;
                 opacity: 0; transition: all 0.3s; }
    /*在鼠标单击之后显示侧边栏菜单项盒子*/
    .myMenu_list_clicked { visibility: visible; opacity: 1;
                         transition: all 0.4s; transition-delay: 0.37s; }
    /*取消列表的默认样式*/
    .myMenu_list ul { list-style-type: none; }
    /*设置侧边栏菜单项的基本样式*/
    .myMenu_list ul li a { float: right; color: white; text-decoration: none;
                         background-color: transparent; margin-top: 11px; }
    /*设置鼠标指针悬浮时侧边栏菜单项的文本颜色*/
    .myMenu_list ul li a:hover {color: green;}
  </style></head>
 <body><div class="myBox"><div id="myMenu" class="myMenu"></div>
  <span class="myLine1"></span><span class="myLine2"></span>
  <span class="myLine3"></span><div class="myMenu_list">
   <ul><li><a href="#">HTML5 专区</a></li><li><a href="#">jQuery 专区</a></li>
       <li><a href="#">CSS3 专区</a></li><li><a href="#">BootStrap 专区</a></li>
       <li><a href="#">AngularJS 专区</a></li></ul></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$('.myMenu').toggleClass('myMenu_clicked')中的 toggleClass()方法是 jQuery 库的方法，用于对设置或移除被选元素(菜单按钮)的一个或多个类进行切换，该方法检查每个元素中指定的类，如果不存在则添加类，如果已设置则删除，这就是所谓的切换效果；transform:scale(200)表示将圆形的菜单按钮放大 200 倍，这远远超出了屏幕的显示范围，但经过盒子的裁剪之后，这个圆形的菜单按钮就变成了矩形的侧边栏，3 条横线

也变成了关闭按钮；transition：all 0.35s ease-in 表示在 350 毫秒内以 ease-in 模式完成侧边栏的显示。

此实例的源文件名是 myHtmlB367.html。

277 使用 transition 属性凸出显示选项卡的当前标签

此实例主要在 transition 中改变选项卡标签的 top 等属性，从而实现在切换选项卡标签时凸出显示当前鼠标选中的标签。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标单击“乐山大佛”标签，则该标签将从所有等高的标签中逐渐向上凸出；如果使用鼠标单击“中央公园”标签，则该标签将从所有等高的标签中逐渐向上凸出，其他标签则恢复为初始状态，如图 277-1 所示；使用鼠标单击“山城夜景”标签将实现类似的功能。有关此实例的主要代码如下。



图 277-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  body { background:black;
        font - family: microsoft yahei, Arial, Helvetica, sans - serif; }
  .myTabs { width: 450px; float: none; list - style: none;
           position: relative; text - align: left; }
  .myTabs li { float: left; display: block;}
  /* 隐藏单选按钮 */
  .myTabs input[type = "radio"] { position: absolute; top: - 9999px; left: - 9999px;}
  /* 设置选项卡标签块的样式 */
  .myTabs label { display: block; padding: 14px 21px; border - radius: 2px 2px 0 0;
    font - size: 20px; font - weight: normal; background: #6A9E52;cursor: pointer;
    position: relative; top: 10px; - webkit - transition: all 0.2s ease - in - out;
    border - top - left - radius: 5px; border - top - right - radius: 5px;
    border - right:solid 1px black;}
  /* 设置鼠标指针悬浮于标签时的样式 */
  .myTabs label:hover { background: #508438; }
```

```
/*设置选项卡内容块的样式*/
.myTabs .myTab-content { z-index: 2; opacity: 0; overflow: hidden; width: 100%;
                          font-size: 17px;line-height: 25px; padding: 25px; position: absolute;
                          top: 53px; left: 0; background: #376B1F; border-radius: 10px;}
/*设置选中标签的样式*/
.myTabs [id^="myTab"]:checked + label {top: 0; padding-top: 17px;
                                        background: #376B1F; }
/*在选中标签时显示内容*/
.myTabs [id^="myTab"]:checked ~ [id^="myTab-content"] { opacity: 1; }
/*设置图像的基本样式*/
img {float: left; border-radius: 10px; padding: 5px;
     width: 210px;height: 135px;box-shadow: 2px 2px 10px black;
     margin-right: 12px; margin-top:17px; margin-bottom: 5px; margin-left:2px; }
</style></head>
<body><div class="content"><ul class="myTabs">
 <li><input type="radio" name="myTabs" id="myTab1" checked/>
  <label for="myTab1">乐山大佛</label>
  <div id="myTab-content1" class="myTab-content">
   <img src="img/B118.jpg" style="float:left;"><p>乐山大佛,地处四川省乐山市,岷江、青衣江和大渡河
三江汇流处,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态匀称,神势肃穆,依
山凿成,临江危坐。大佛通高71米,头高14.7米,头宽10米,发髻1051个,耳长6.7米,鼻和眉长5.6米,嘴巴
和眼长3.3米,颈高3米,肩宽24米,手指长8.3米,从膝盖到脚背28米,脚背宽9米,脚面可围坐百人以上。
</p></div></li>
  <li><input type="radio" name="myTabs" id="myTab2"/>
  <label for="myTab2">中央公园</label>
   <div id="myTab-content2" class="myTab-content">
   <img src="img/B179A.jpg" style="float:left;"><p>重庆中央公园是西南地区最大的开放式城市中心
公园,西距在建的悦来重庆国际会展城3公里,东距江北国际机场5公里。它是一个依托重庆山水风貌特色,融
会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一起,成为两江新区重大
城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲去处,重庆的标志性公
园。</p></div></li>
  <li><input type="radio" name="myTabs" id="myTab3"/>
   <label for="myTab3">山城夜景</label>
   <div id="myTab-content3" class="myTab-content">
   <img src="img/B179B.jpg" style="float:left;"><p>山城夜景即重庆夜景。初夜山城,以繁华区灯饰群
为中心,干道和桥梁华灯为纽带,万家民居灯火为背景,层见叠出,构成一片高下井然、错落有致、远近互衬的灯
的海洋。更兼两江波澄银树,浪卷金花,满天繁星似人间灯火,遍地华灯若天河群星,上下浑然一体,五彩交相辉
映,如梦如幻,如诗如歌,堪足撩人耳目,动人心旌。不览夜景,未到重庆。</p></div></li></ul></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.myTabs [id^="myTab"]:checked中的"[id^="myTab"]"是一个属性选择器,原型是[attribute^=value],表示选择器匹配属性值以指定值(例如myTab)开头的每个元素;-webkit-transition: all 0.2s ease-in-out表示在200毫秒内以ease-in-out模式完成凸出显示当前标签的动作。

此实例的源文件名是myHtmlB374.html。

278 使用关键帧动画定制移动和旋转的图像

此实例主要通过使用关键帧动画定制移动和旋转的图像。当在Google Chrome浏览器中显示该页面时,摄像机图像初始位于左上角,当鼠标指针悬浮在摄像机图像上时,则摄像机图像执行旋转和移动的动作,如图278-1所示。有关此实例的主要代码如下。

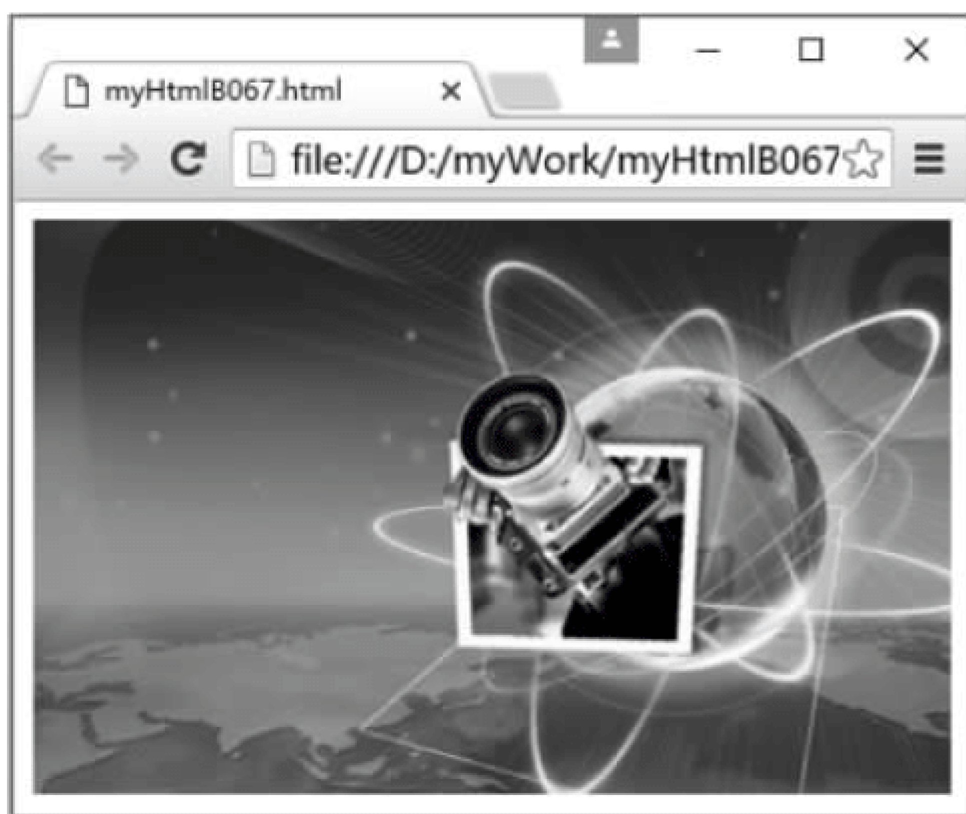


图　278-1

```
<! doctype html >< html >< head >< meta charset = "UTF - 8">
 < style type = "text/css">
  div {background - image: url("img/a050A.jpg");width: 400px; height: 250px;}
  img { position: absolute; top: 10px;left: 20px;}
   @keyframes myImage { 0 % {top: 10px;left: 20px;transform: rotate(0deg);}
                        40 % {top: 50px; left: 90px; transform: rotate(130deg); }
                        70 % {top: 90px; left: 220px;transform: rotate(220deg); }
                        100 % {top: 120px;left: 260px;transform: rotate(360deg);} }
   div:hover img {animation - name:myImage; animation - duration: 5s;
                  animation - timing - function: linear;}
 </style></head>
<body><div>< img src = "img/B067A.png"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation-timing-function 用于设置动画的执行过程中的速度变化模式为 linear，animation-name 用于设置动画名称为 myImage，animation-duration 用于设置动画的持续时间为 5 秒。@keyframes 用于定义关键帧集合中的每个关键帧上的状态。除了 animation-timing-function、animation-name、animation-duration 几个常用属性以外，关键帧动画还可以使用下列属性。

(1) animation-delay：用于指定延迟多少秒或多少毫秒后开始执行动画。

(2) animation-iteration-count：用于指定动画的执行次数，可指定为 infinite(无限次)。

(3) animation-direction：用于指定动画的执行方向，可指定的属性值有 normal(动画执行完毕后返回初始状态)、alternate(交替更改动画的执行方向)、reverse(反方向执行动画)、alternate-reverse(从反方向开始交替更改动画的执行方向)。

此实例的源文件名是 myHtmlB067.html。

279　使用关键帧动画实现闪烁的阴影光圈

此实例主要在关键帧动画的 animation 属性中设置 infinite(无限次)属性值，从而实现在元素的外围创建不断闪烁的阴影光圈。当在 Google Chrome 浏览器中显示该页面时，黄色太阳的周围有个不断变换的红色阴影在不停闪烁，如图 279-1 所示。有关此实例的主要代码如下。

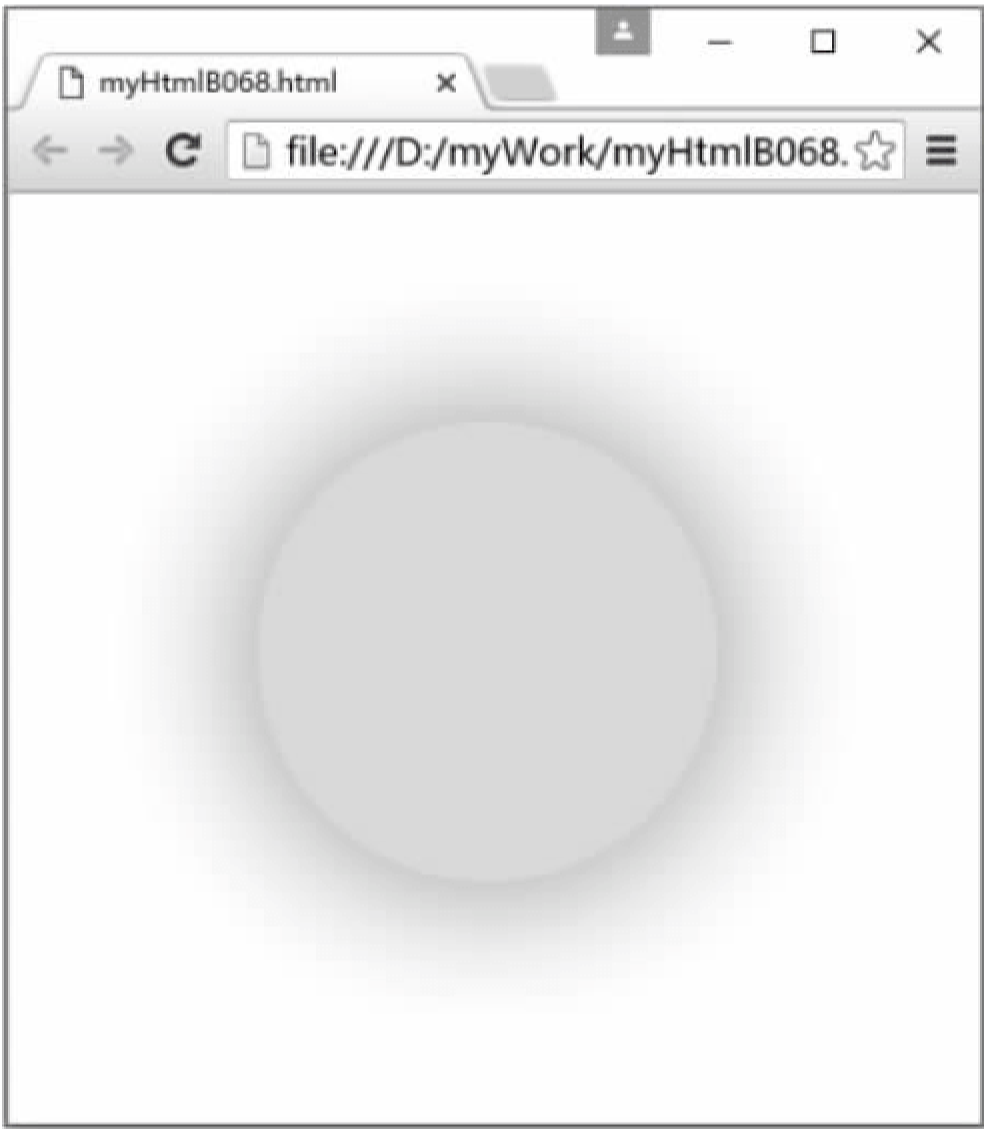


图 279-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { width: 200px; height: 200px; margin:100px;
       border-radius: 100px; box-shadow: 0 0 10px rgba(204, 102, 0, 0.8);
       background: yellow;
       animation: myFlash 1.5s ease infinite;}
  @keyframes myFlash { 0% { box-shadow: 0 0 50px rgba(255,0,0, 0.1); }
                      50% { box-shadow: 0 0 100px rgba(255,0,0, 0.3);}
                      100% {box-shadow: 0 0 150px rgba(255,0,0,0.8); } }
 </style></head>
<body><div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，@keyframes myFlash 用于设置动画所需要的 myFlash 关键帧集合中的多个关键帧；animation：myFlash 1.5s ease infinite 表示采用 ease 模式在 1.5 秒内执行完成关键帧动画集合 myFlash 中的所有关键帧，并且永不停止（infinite）。animation 属性是一个简写属性，用于设置 6 个动画属性，即 animation-name、animation-duration、animation-timing-function、animation-delay、animation-iteration-count、animation-direction，该属性的语法格式如下。

```
animation: name duration timing-function delay iteration-count direction
```

animation 属性值的说明如表 279-1 所示。

表 279-1 animation 属性值及说明

属 性 值	描 述
animation-name	规定需要绑定到选择器的 keyframe 名称
animation-duration	规定完成动画所花费的时间，以秒或毫秒计
animation-timing-function	规定动画的速度曲线

续表

属　性　值	描　　述
animation-delay	规定动画开始之前的延迟，以秒或毫秒计
animation-iteration-count	规定动画应该播放的次数
animation-direction	规定是否应该轮流反向播放动画

animation-timing-function 使用名为三次贝塞尔（Cubic Bezier）函数的数学函数来生成速度曲线，用户可以在该函数中使用自己的值，也可以使用预定义值，例如 linear（动画从头到尾的速度是相同的）、ease（默认值，动画以低速开始，然后加快，在结束前变慢）、ease-in（动画以低速开始）、ease-out（动画以低速结束）、ease-in-out（动画以低速开始和结束）。cubic-bezier（n，n，n，n）表示在 cubic-bezier 函数中使用自定义值，可以是从 0 到 1 的数值。

animation-iteration-count 属性定义动画的播放次数，该属性值可以是具体的数字，也可以使用 infinite 规定动画无限次播放。

animation-direction 属性定义是否应该轮流反向播放动画。如果该属性值是 alternate，则动画会在奇数次（1、3、5 等）正常播放，而在偶数次（2、4、6 等）反向播放。如果该属性值是 normal，即默认值，动画正常播放。

此实例的源文件名是 myHtmlB068.html。

280　使用关键帧动画实现以淡入效果显示图像

此实例主要通过关键帧动画控制元素（img）的不透明度（opacity）来实现以淡入效果显示图像。当在 Google Chrome 浏览器中显示该页面时，图像将在 5 秒内以由浅入深的淡入效果显示出来，如图 280-1 所示。有关此实例的主要代码如下。



图　280-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  @keyframes fadeIn { 0% { /* 0.0 (完全透明) */ opacity: 0; }
                     100% {/* 1.0 (完全不透明) */ opacity: 1;} }
  img {animation-name: fadeIn; animation-duration: 5s; border-radius: 15px;
       animation-timing-function: linear; animation-iteration-count: 1; }
```

```
</style></head>
<body><img src="img/B069.jpg"/></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,opacity：0 表示完全透明,看不见图像；opacity：1 表示完全不透明,即完全显示图像。当加载网页时,浏览器将根据设置的关键帧动画属性(animation-timing-function：linear 表示以均匀速度改变不透明度,animation-iteration-count：1 表示此动画仅执行一次,animation-duration：5s 表示在 5 秒内完成动画)以由浅入深的淡入效果显示图像。

此实例的源文件名是 myHtmlB069.html。

281 使用关键帧动画实现以卷帘效果显示图像

此实例主要通过关键帧动画控制元素(img)的顶端坐标 top,从而实现以卷帘效果显示图像。当在 Google Chrome 浏览器中显示该页面时,图像将在 5 秒内以从上向下的卷帘效果显示出来,如图 281-1 所示。有关此实例的主要代码如下。



图 281-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   @keyframes slideDown{ 0% { top: -350px; } 100% { top: 0px;} }
   img {animation-name: slideDown;animation-duration: 5s;
        animation-timing-function: linear; animation-iteration-count: 1;
        border-radius: 15px;width:300px;height:350px; position: absolute;}
 </style></head>
<body><div><img src="img/B070.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,top：－350px 表示图像在浏览器之外(因为图像的高度是 350px),因此用户看不见图像；top：0px 表示图像在浏览器的顶端,即完全显

示图像。当加载网页时，浏览器将根据设置的关键帧动画属性(animation-timing-function：linear 表示以匀速改变 top 值，animation-iteration-count：1 表示此动画仅执行一次，animation-duration：5s 表示在 5 秒内完成动画)来实现以从上向下的卷帘效果显示图像。

此实例的源文件名是 myHtmlB070.html。

282 使用关键帧动画实现从左向右滑入图像

此实例主要通过关键帧动画控制元素(img)的左端坐标 left，从而实现以从左到右的抽屉效果滑入图像。当在 Google Chrome 浏览器中显示该页面时，单击"从左向右滑入图像"按钮，则将在 5 秒内以从左向右的抽屉效果滑入图像，如图 282-1 所示。有关此实例的主要代码如下。



图 282-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnIn").click(function() { //从左向右滑入图像
    $("img").css("display", "block");
    $("img").css("animation-name", "slideIn");
  });});
 </script><style type="text/css">
  @keyframes slideIn { 0% {left: -390px;} 100% {left: 0px; } }
  img {animation-duration: 5s; animation-timing-function: linear;
       animation-iteration-count: 1; width: 390px;height: 250px;
       position: absolute; display: none;}
 </style></head>
<body><p><input type="button" value="从左向右滑入图像" id="myBtnIn" style="width:390px"></p>
<div><img src="img/B071.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，left：－390px 表示图像在浏览器左端之外(因为图像的宽度是 390px)，因此用户看不见图像；left：0px 表示图像在浏览器的左端，即完全显示图像。每个关键帧动画必须设置动画名称 animation-name，否则动画不被执行。此实例通过 jQuery 代码"$("img").css("animation-name", "slideIn")"设置动画名称，因此在单击按钮后浏览器将根据关键帧动画属性(animation- timing-function：linear 表示匀速改变 left 值，animation-

iteration-count：1 表示此动画仅执行一次，animation-duration：5s 表示在 5 秒内完成动画）来实现以从左向右的滑入效果显示图像。

此实例的源文件名是 myHtmlB071.html。

283 使用关键帧动画高仿抽奖转盘的转动特效

此实例主要通过关键帧动画控制元素（img）的 transform 属性指定的 rotate()方法，从而实现高仿抽奖转盘的转动特效。当在 Google Chrome 浏览器中显示该页面时，“开始抽奖”指针将围绕中心变速旋转，如图 283-1 所示。有关此实例的主要代码如下。



图 283-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  @keyframes myPendulum { 0% { transform: rotate(0deg); }
                          100% {transform: rotate(360deg); } }
  img { animation-name: myPendulum; animation-duration: 2s;
        animation-timing-function: ease; animation-iteration-count: infinite;
        width: 200px; height: 200px; top: 110px; left: 110px; position: absolute; }
  div { width: 400px;height: 400px; background-image: url("img/B072A.png");
        background-size: 100% 100%; }
 </style></head>
<body><div><img src="img/B072B.png"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，0% { transform：rotate(0deg)表示最初不执行旋转，100% {transform：rotate(360deg)表示旋转 360°；animation-duration：2s 表示此动画旋转一次的时间是两秒，animation-timing- function：ease 表示在两秒内此动画按照 ease(动画以低速开始，然后加快，在结束前变慢）模式执行旋转，animation-iteration-count：infinite 表示旋转一周后重新开始，并且永不结束。

此实例的源文件名是 myHtmlB072.html。

284　使用关键帧动画高仿弹簧的拉伸、压缩效果

此实例主要通过关键帧动画控制元素(img)的 transform 属性指定的 scaleX()方法,从而实现在 X 轴方向上拉伸和压缩弹簧的效果。当在 Google Chrome 浏览器中显示该页面时,弹簧将自动在 X 轴方向上拉伸和压缩,如图 284-1 所示。有关此实例的主要代码如下。

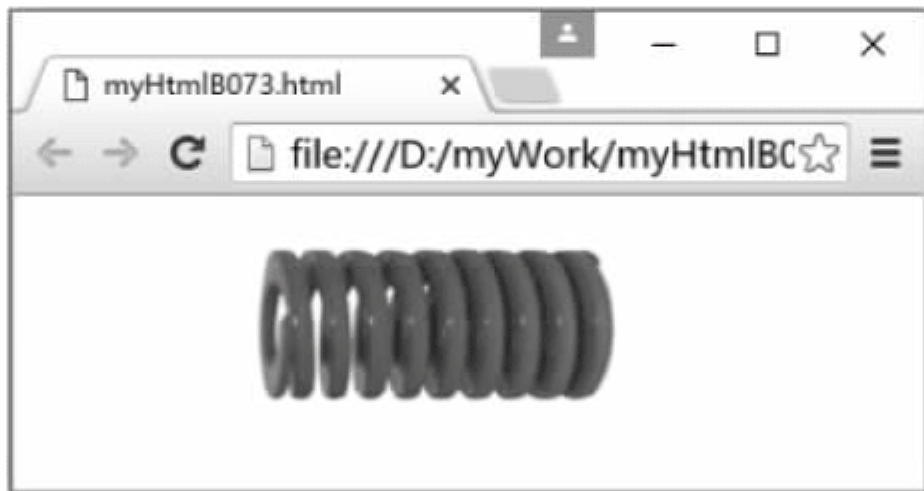


图　284-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  @keyframes myScaleX { 0% { transform: scaleX(1.0);}
                        100% { transform: scaleX(0.7); } }
  img {animation-name: myScaleX; animation-duration: 2s;
       animation-timing-function: ease; animation-iteration-count: infinite;
       animation-direction:alternate;}
  div{ display: table-cell; vertical-align: middle;
       text-align: center; width:350px; height:100px; }
 </style></head>
<body><div><img src="img/B073.jpg"/></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,0%{transform: scaleX(1.0);}表示开始压缩时的(弹簧的)原始长度,100%{transform: scaleX(0.7);}表示结束压缩时的(弹簧的)长度; animation-duration: 2s 表示此动画压缩(或拉伸)一次的时间是两秒,animation-timing-function: ease 表示在两秒内此动画按照 ease(动画以低速开始,然后加快,在结束前变慢)模式执行拉伸和压缩,animation-iteration-count: infinite 表示拉伸和压缩的动画永不结束,animation-direction: alternate 表示拉伸和压缩动画交替进行。

此实例的源文件名是 myHtmlB073.html。

285　使用关键帧动画实现多个图像自动轮播

此实例主要通过在 animation 属性中设置关键帧动画实现多个图像的自动轮播。当在 Google Chrome 浏览器中显示该页面时将自动无限循环地轮播 4 幅图像,当显示第一幅图像时,图像与右上角的索引的显示效果如图 285-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  /*设置轮播图像的关键帧*/
```

```
@-webkit-keyframes slide { from, to { top: 0; }
                          12.5% { top: 0; } 25% { top: -375px; }
                          37.5% { top: -375px; } 50% { top: -750px; }
                          62.5% { top: -750px; } 75% { top: -1125px;}
                          87.5% { top: -1125px;} }
/*设置轮播索引的关键帧*/
@-webkit-keyframes index_1 { from, 25%, to { background-color: rgba(0, 0, 0, 0.5); }
                            0% { background-color: rgba(255, 0, 0, 0.5); } }
@-webkit-keyframes index_2 {from, 50%, to { background-color: rgba(0, 0, 0, 0.5); }
                            25% { background-color: rgba(255, 0, 0, 0.5); } }
@-webkit-keyframes index_3 { from, 75%, to { background-color: rgba(0, 0, 0, 0.5); }
                            50% { background-color: rgba(255, 0, 0, 0.5); } }
@-webkit-keyframes index_4 {from, 100%, to { background-color: rgba(0, 0, 0, 0.5);}
                            75% { background-color: rgba(255, 0, 0, 0.5); } }
/*设置图像框的基本样式*/
.box { width: 500px; height: 375px; margin: 1em auto;
       position: relative; overflow: hidden; border-radius: 10px;
       box-shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset; }
/*设置图像的样式*/
img { vertical-align: top; height: 375px; width: 500px; }
/*设置无序列表的样式*/
.list { margin: 0; padding: 0; position: absolute;
        -webkit-animation: slide 20s infinite; }
/*设置索引栏位于右上角位置*/
.index { position: absolute; right: 10px; top: 10px; }
/*设置每个索引框的基本样式*/
.index a { display: inline-block; width: 30px; height: 30px; line-height: 30px;
           margin-left: .5em; border-radius: 10px; background-color: rgba(0, 0, 0, 0.5);
           text-align: center; text-decoration: none !important;
           color: yellow; font-size: 20px; font-weight: bold; }
/*设置4个索引框轮播动画*/
.index_1 { -webkit-animation: index_1 20s step-end infinite;}
.index_2 { -webkit-animation: index_2 20s step-end infinite;}
.index_3 { -webkit-animation: index_3 20s step-end infinite;}
.index_4 { -webkit-animation: index_4 20s step-end infinite;}
</style></head>
<body><div class="box">
 <ul class="list">
  <li><img src="img/B180A.jpg"/></li><li><img src="img/B180B.jpg"/></li>
  <li><img src="img/B180C.jpg"/></li><li><img src="img/B180D.jpg"/></li></ul>
 <div class="index"><a href="javascript:" class="index_1">1</a>
  <a href="javascript:" class="index_2">2</a>
  <a href="javascript:" class="index_3">3</a>
  <a href="javascript:" class="index_4">4</a></div>
</div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,@-webkit-keyframes slide 用于设置轮播图像的多个关键帧的纵向位置,在轮播之前不可见,但已经用负数定位;-webkit-animation: slide 20s infinite 表示采用 infinite 模式,在 20 秒内执行完成所有关键帧。animation 属性是一个简写属性,用于设置 6 个动画属性,即 animation-name、animation-duration、animation-timing-function、animation-iteration-count、animation-delay、animation-direction。

此实例的源文件名是 myHtmlB181.html。

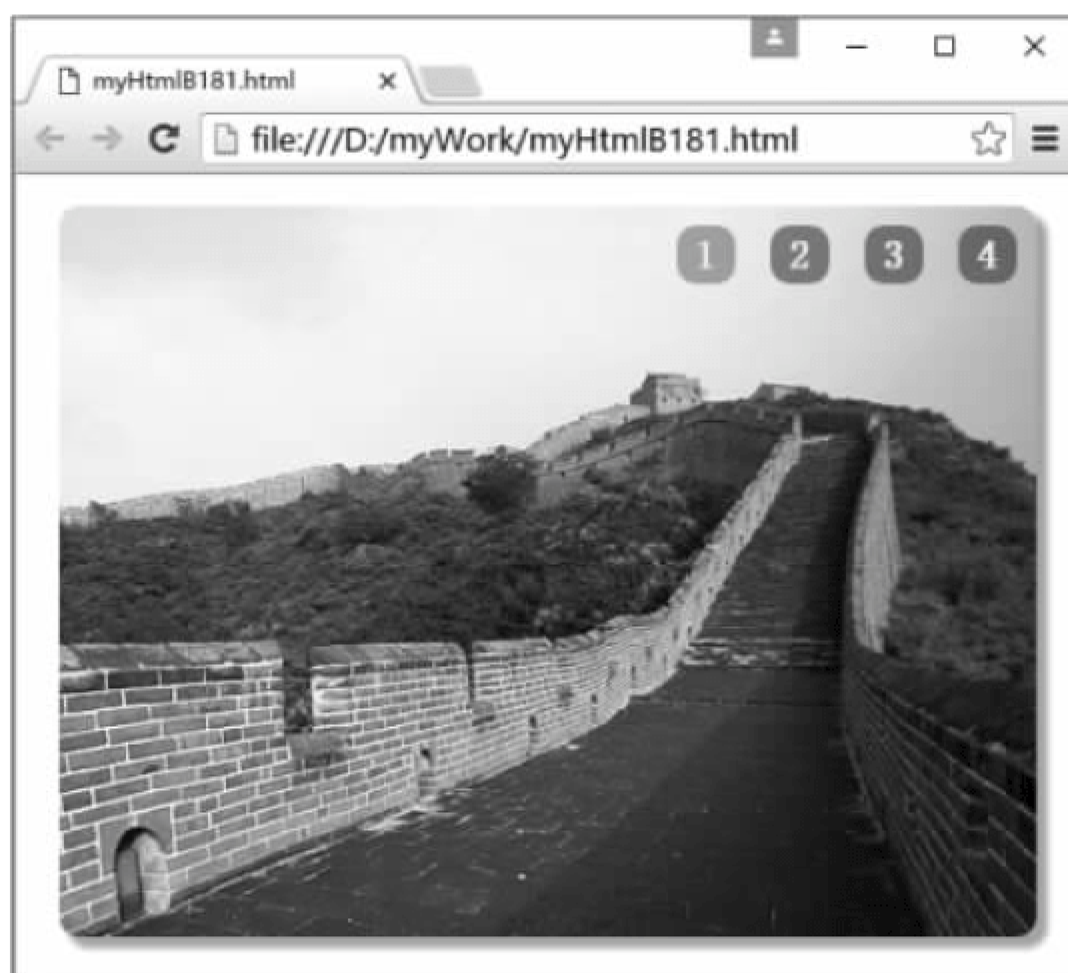


图　285-1

286　使用关键帧动画实现圆环转圈更新特效

此实例主要通过关键帧动画控制元素的 transform 属性指定的 rotate()方法，从而实现圆环的转圈更新特效。当在 Google Chrome 浏览器中显示该页面时，黄色的圆环将按照顺时针方向不断转圈，如图 286-1 所示。有关此实例的主要代码如下。

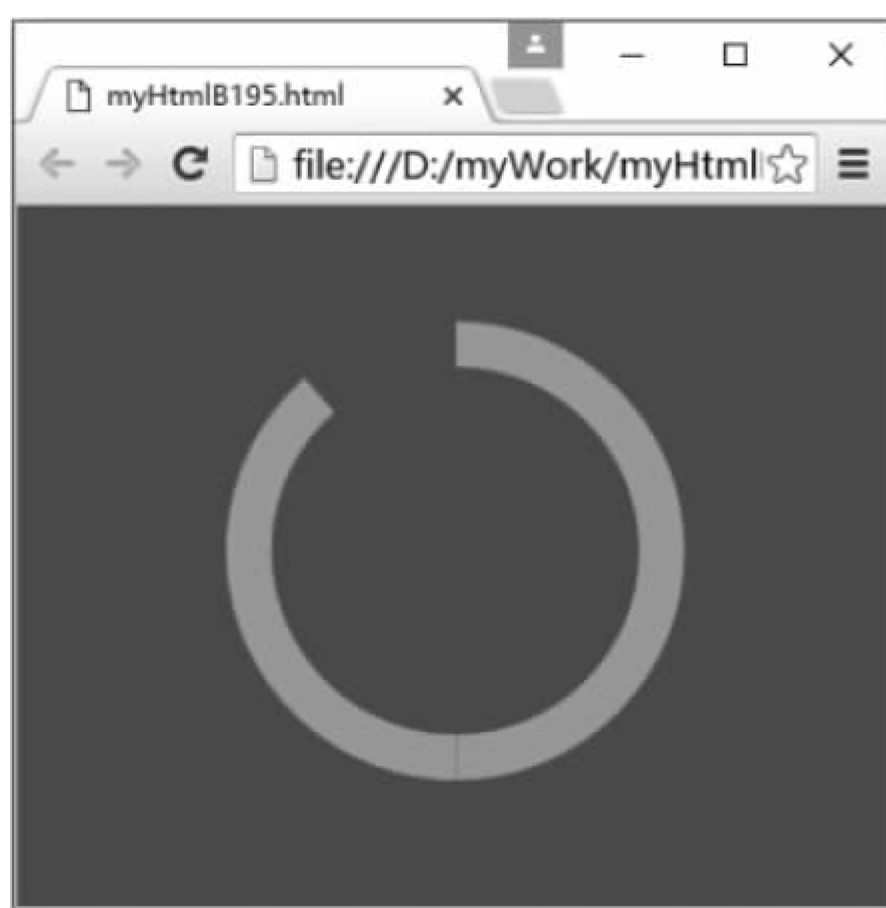


图　286-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body { background-color: green; }
  .box { width: 200px; height: 200px; margin: 50px auto;
         position: relative; border: 0px solid #DDD; }
```

```
    .left { left: 0; }
    .right { right: 0; }
    .area { width: 100px; height: 200px; position: absolute;
           top: 0; overflow: hidden; }
    .progress { width: 160px; height: 160px; border: 20px solid transparent;
               border-radius: 50%; position: absolute; top: 0; }
    .leftcircle { border-bottom: 20px solid orange; border-left: 20px solid orange;
                 left: 0; -webkit-animation: loading_left 5s linear infinite; }
    @-webkit-keyframes loading_left { 0%, 50% { -webkit-transform: rotate(-135deg);}
                                      100% { -webkit-transform: rotate(45deg);} }
    .rightcircle { border-top: 20px solid orange; border-right: 20px solid orange;
                  right: 0; -webkit-animation: loading_right 5s linear infinite; }
    @-webkit-keyframes loading_right { 0% { -webkit-transform: rotate(-135deg); }
                                       50%, 100% { -webkit-transform: rotate(45deg); }}
   </style></head>
  <body><div class="box">
   <div class="area left"><div class="progress leftcircle"></div></div>
   <div class="area right"><div class="progress rightcircle"></div></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，overflow：hidden 表示在内容超出范围时超出部分是不可见的；-webkit-transform：rotate(45deg)表示旋转 45°；-webkit-animation：loading_right 5s linear infinite 表示在 5 秒内以线性方式完成关键帧动画 loading_right，infinite 表示完成一次关键帧动画 loading_right 后重新开始，并且永不结束。在此实例中，整个动画实际上是由左、右两部分拼接而成的，并通过正、负角度来控制旋转。

此实例的源文件名是 myHtmlB195.html。

287 使用关键帧动画实现淡入、淡出轮播特效

此实例主要通过关键帧动画控制元素的 opacity 属性实现淡入、淡出的轮播特效。当在 Google Chrome 浏览器中显示该页面时，轮播 4 幅图像的部分效果如图 287-1 所示。有关此实例的主要代码如下。



图 287-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
    * { margin: 0; padding: 0; }
    .myBox { width: 300px; height: 200px; box-shadow: 1px 1px 10px #AAA;
             padding: 5px; margin: 20px auto; position: relative; }
    ul { width: 300px; height: 200px; position: relative;overflow: hidden;}
    li { list-style: none; position: absolute; top: 0px; left: 0; }
    img { width: 300px; height: 200px; }
    #first { -webkit-animation: myImg1 10s infinite; }
    #second { -webkit-animation: myImg2 10s infinite;}
    #third { -webkit-animation: myImg3 10s infinite; }
    #fourth { -webkit-animation: myImg4 10s infinite; }
    @-webkit-keyframes myImg1 { 0% { opacity: 1; } 15% { opacity: 1; }
                               25% { opacity: 0; } 90% { opacity: 0; } 100% {opacity: 1; } }
    @-webkit-keyframes myImg2 { 0% { opacity: 0; } 15% { opacity: 0;}
                               25% { opacity: 1; } 40% { opacity: 1; }
                               50% { opacity: 0; } 100% { opacity: 0; }}
    @-webkit-keyframes myImg3 { 0% { opacity: 0; } 40% { opacity: 0; }
                               50% { opacity: 1; } 65% { opacity: 1; }
                               75% { opacity: 0; } 100% { opacity: 0; } }
    @-webkit-keyframes myImg4 { 0% { opacity: 0; } 65% { opacity: 0; }
                               75% { opacity: 1; } 90% { opacity: 1; } 100% { opacity: 0; } }
  </style></head>
 <body><div class="myBox"><ul><li id="first"><img src="img/B151A.jpg"/></li>
  <li id="second"><img src="img/B151B.jpg"/></li>
  <li id="third"><img src="img/B152A.jpg"/></li>
  <li id="fourth"><img src="img/B152B.jpg"/></li> </ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，opacity 属性用于设置元素的不透明级别，即从 0.0（完全透明）到 1.0（完全不透明）；-webkit-animation: myImg1 10s infinite 表示在 10 秒内执行 myImg1 关键帧指定的动作，并且在完成动画后重新开始，永不结束。此实例最精华的思想是在 10 秒这个时间轴上 4 个关键帧动画的每个百分点均保证不冲突，例如在 myImg1 关键帧中，在 0%、15%、100%这几个点位上显示第一幅图像（opacity: 1）时其他 3 个关键帧的 0%、15%、100%这几个点位绝不允许出现相同的属性值，即 opacity 属性必须为 0，其余以此类推，因此如果要显示更多的图像，只需要在保证不冲突的前提下依次修改各个关键帧的点位（百分比值）的 opacity 属性即可。

此实例的源文件名是 myHtmlB201.html。

288 使用关键帧动画实现文字水平滚动显示

此实例主要通过在关键帧动画中控制元素的 left 属性实现文字的水平滚动显示。当在 Google Chrome 浏览器中显示该页面时，“炫酷应用实例集锦”的水平滚动显示效果如图 288-1 所示。有关此实例的主要代码如下。



图 288-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { width: 400px; position: absolute; overflow: hidden; color: black;
          /* 创建阴影特效文字 */
          text-shadow: 0 0 5px #FF0000; font-size: 40px;
          font-weight: bold; font-family: 宋体; }
  .myMarquee { position: relative; -webkit-animation: marquee 10s linear infinite; }
  @-webkit-keyframes marquee { 0% { left: -100%; } 100% { left: 100%; } }
 </style></head>
<body><div class="myBox"><p class="myMarquee">炫酷应用实例集锦</p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，left 属性规定元素的左边缘，如果元素的 position 属性值为 static，那么设置 left 属性不会产生任何效果；left 属性可以用%设置以包含元素的百分比计算的左边位置，可使用负值；用户也可以使用 px、cm 等单位设置元素的左边位置，可使用负值。在此实例的关键帧中，当 left：－100%时演示文字是不可见的。-webkit-animation：marquee 10s linear infinite 表示在 10 秒内以 linear 模式执行 marquee 关键帧指定的动作，并且在完成动画后重新开始，永不结束。

此实例的源文件名是 myHtmlB203.html。

289 使用关键帧动画高仿小圆点的加载状态

此实例主要通过在关键帧动画中旋转或隐藏元素高仿 Windows 10 操作系统启动时的小圆点加载进度状态。当在 Google Chrome 浏览器中显示该页面时，小圆点转圈的加载进度状态显示效果如图 289-1 所示。有关此实例的主要代码如下。



图 289-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .loading { margin: 50px 160px; }
  .loading .dot { position: absolute; opacity: 0; width: 40px; height: 40px;
         transform: rotate(225deg); animation-name: loading;
          animation-iteration-count: infinite;animation-duration: 5.28s; }
  .loading .dot:after { content: ""; position: absolute; background: green;
                         width: 6px; height: 6px; border-radius: 50%;}
  .loading .dot:nth-child(2) { animation-delay: 0.23s; }
  .loading .dot:nth-child(3) { animation-delay: 0.46s; }
  .loading .dot:nth-child(4) { animation-delay: 0.69s; }
```

```
.loading .dot:nth-child(5) { animation-delay: 0.92s; }
@keyframes loading { 0% {transform: rotate(225deg); opacity: 1;
        animation-timing-function: cubic-bezier(0, 0, 0.58, 1.0); }
    8% {transform: rotate(345deg);
        animation-timing-function: cubic-bezier(0.0, 0.0, 1.0, 1.0);}
   30% {transform: rotate(455deg);
        animation-timing-function: cubic-bezier(0.42, 0, 0.58, 1.0);}
   40% {transform: rotate(690deg);
        animation-timing-function: cubic-bezier(0.0, 0.0, 1.0, 1.0);}
   60% {transform: rotate(815deg); opacity: 1;
        animation-timing-function: cubic-bezier(0, 0, 0.58, 1.0);}
   75% {transform: rotate(965deg);
        animation-timing-function: cubic-bezier(0, 0, 0.58, 1.0);}
   76% {opacity: 0; }
   100% {opacity: 0; }}
</style></head>
<body><div class="loading">
    <div class="dot"></div><div class="dot"></div><div class="dot"></div>
    <div class="dot"></div><div class="dot"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,transform: rotate(225deg)表示将小圆点(6,6)所在的小圆圈(40,40)旋转225°;animation-duration: 5.28s定义关键帧动画完成一个周期所需要的时间是5.28秒;animation-iteration-count: infinite规定关键帧动画应该无限次播放;animation-name: loading表示将要执行的是loading关键帧动画;animation-delay: 0.23s定义关键帧动画等待的时间是230毫秒;opacity: 1表示显示小圆点,opacity: 0表示隐藏小圆点;animation-timing-function: cubic- bezier(0, 0, 0.58, 1.0)规定动画的速度变化曲线是cubic-bezier(0, 0, 0.58, 1.0)。

此实例的源文件名是myHtmlB204.html。

290 使用关键帧动画实现在单行中轮播文本

此实例主要通过关键帧动画控制元素的opacity属性,从而实现在单行中逐行轮播文本。当在Google Chrome浏览器中显示该页面时,轮播4行文本的部分效果如图290-1所示。有关此实例的主要代码如下。



图 290-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
    li { list-style-type: none; position: absolute;
        /* 创建阴影特效文字 */
        text-shadow: 5px 5px 5px gray; color: navy;font-size: 30px;
        font-weight: bold;font-family: 宋体;}
    .myBox { margin: 50px auto; }
    #first { -webkit-animation: p_1 10s infinite; }
    #second { -webkit-animation: p_2 10s infinite; }
    #third { -webkit-animation: p_3 10s infinite;}
    #fourth { -webkit-animation: p_4 10s infinite; }
    @-webkit-keyframes p_1 { 0% { opacity: 1; } 15% { opacity: 1; }
        25% { opacity: 0; } 90% { opacity: 0; } 100% {opacity: 1; } }
    @-webkit-keyframes p_2 { 0% { opacity: 0; } 15% { opacity: 0; }
        25% { opacity: 1; } 40% { opacity: 1; }
        50% { opacity: 0; } 100% { opacity: 0; }}
    @-webkit-keyframes p_3 { 0% { opacity: 0; } 40% { opacity: 0; }
        50% { opacity: 1; } 65% { opacity: 1; }
        75% { opacity: 0; } 100% { opacity: 0; } }
    @-webkit-keyframes p_4 { 0% { opacity: 0; } 65% { opacity: 0; }
        75% { opacity: 1; } 90% { opacity: 1; } 100% { opacity: 0;} }
  </style></head>
<body><div class="myBox">
 <ul><li id="first">孤山寺北贾亭西,水面初平云脚低。</li>
     <li id="second">几处早莺争暖树,谁家新燕啄春泥。</li>
     <li id="third">乱花渐欲迷人眼,浅草才能没马蹄。</li>
     <li id="fourth">最爱湖东行不足,绿杨阴里白沙堤。</li></ul></div> </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，opacity 属性用于设置元素的不透明级别，即从 0.0(完全透明)到 1.0(完全不透明)，opacity：1 表示显示文本，opacity：0 表示隐藏文本；-webkit-animation：p_1 10s infinite 表示在 10 秒内执行 p_1 关键帧指定的动作，并且在完成动画后重新开始，永不结束。此实例最精华的思想是在 10 秒这个时间轴上 4 个关键帧动画的每个百分点均保证不冲突，例如在@-webkit- keyframes p_1 关键帧中，在 0%、15%、100%这几个点位上显示第一行文本(opacity：1)时其他 3 个关键帧的 0%、15%、100%这几个点位绝不允许出现相同的属性值，即 opacity 属性必须为 0，其余以此类推，因此如果要显示更多行的文本，只需要在保证不冲突的前提下依次修改各个关键帧的点位(百分比值)的 opacity 属性即可。

此实例的源文件名是 myHtmlB205.html。

291 使用关键帧动画高仿白云在天空中游走

此实例主要通过关键帧动画控制 img 元素的 top、left 属性，从而高仿白云在天空中游走的动画效果。当在 Google Chrome 浏览器中显示该页面时，在图 291-1 的红色圆圈中的白云(它实际上是一幅 png 图像)将根据关键帧指定的位置从左向右游走，并且周而复始，永不停歇。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  html { background-image: url(img/B214B.jpg);background-size: 100% auto; }
  img { width:90px; height:60px; position: absolute;
       animation: myTrack 20s ease-in-out infinite; }
```

```
  @-webkit-keyframes myTrack { 0% { left: -10%; top:0px; }
      10%{ left:10%; top:70px;} 20%{ top:10px;} 50%{ top:65px;}
      70%{ top:24px;}            80%{ top:70px;} 100% {left: 99%;} }
 </style></head>
<body><img src="img/B214.png"/></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation: myTrack 20s ease-in-out infinite 表示在 20 秒内以 ease-in-out 模式执行 myTrack 定义的关键帧动画，并且周而复始，永不停歇。myTrack 的 100%是相对于时间轴而言的，比如在此实例中，20%是指在 20×20%=4 秒时游走的白云应该定位在 top 值为 10px 的位置。

此实例的源文件名是 myHtmlB214.html。

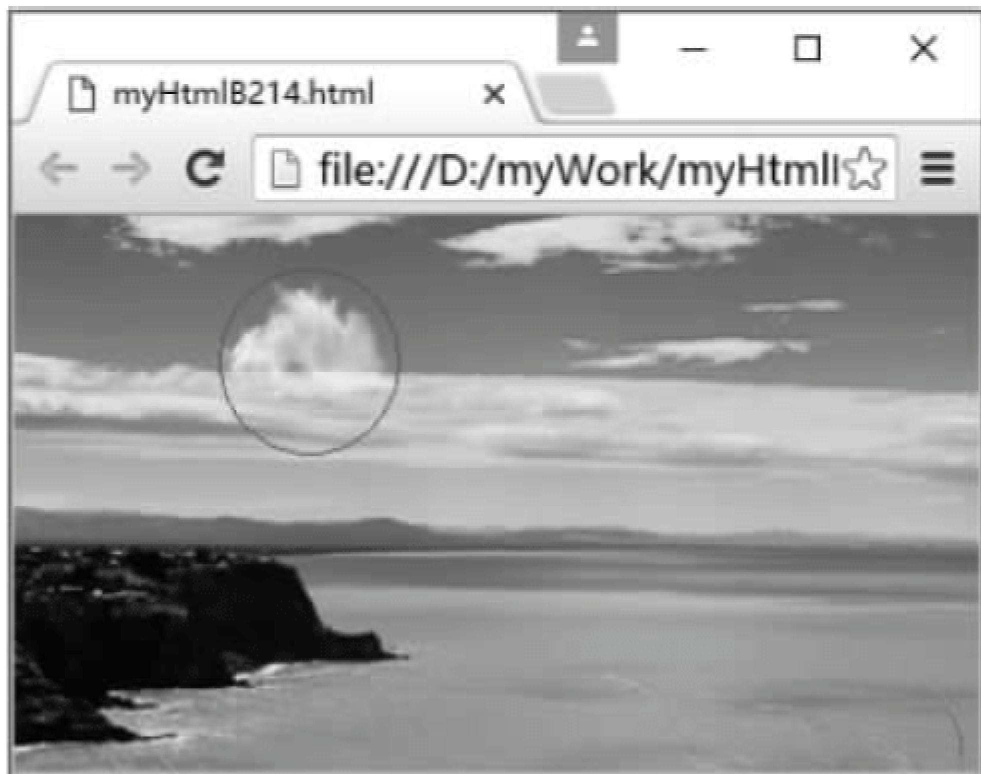


图 291-1

292 使用关键帧动画实现在按钮上扩散波纹

此实例主要通过在关键帧动画中设置 transform 属性值为 scale(2)将元素放大两倍，从而实现在单击按钮时显示扩散的波纹。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标单击按钮，则将以鼠标单击点为中心显示扩散的浅灰色波纹，如图 292-1 所示。有关此实例的主要代码如下。



图 292-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script>
   $(function() {
    $("button").on('click', function(e) {
      var target = e.target;
      var rect = target.getBoundingClientRect();
      var ripple = target.querySelector('.ripple');
```

```
        if (!ripple) {//创建 ripple
          ripple = document.createElement('span');
          ripple.className = 'ripple';
          ripple.style.height = ripple.style.width = Math.max(rect.width, rect.height) + 'px';
          target.appendChild(ripple);
         }
         //显示 ripple 扩散特效
         ripple.classList.remove('show');
         var top = e.pageY - rect.top - ripple.offsetHeight/2 - document.body.scrollTop;
         var left = e.pageX - rect.left - ripple.offsetWidth/2 - document.body.scrollLeft;
         ripple.style.top = top + 'px';
         ripple.style.left = left + 'px';
         ripple.classList.add('show');
   });});
  </script>
  <style type = "text/css">
    /* 裁剪按钮之外的扩散圆 */
    button { position: relative;width: 13em; height: 3em; overflow: hidden; }
    .ripple { position: absolute; background: lightgray;
              border - radius: 100 % ; transform: scale(0); }
    .ripple.show { animation: myRipple 1s ease - out;}
    /* 将扩散圆放大两倍 */
    @ - webkit - keyframes myRipple {100 % { transform: scale(2); opacity: 0; } }
  </style></head>
 <body><center><button>点我试试看</button></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,animation: myRipple 1s ease-out 表示在 1 秒的时间内以 ease-out 模式执行 myRipple 定义的关键帧动画。myRipple 的 100%是相对于时间轴而言的,比如在此实例中波纹扩散时间只有 1 秒。由于波纹特效是在用鼠标单击按钮时使用 jQuery 代码动态添加的,为了保证多次单击均有效,必须先 ripple.classList.remove('show'),然后再 ripple.classList.add('show')。

此实例的源文件名是 myHtmlB220.html。

293 使用关键帧动画高仿因信号干扰花屏的特效

此实例主要通过在关键帧动画中设置 clip 属性裁剪图形,从而高仿因信号干扰花屏的特效。当在 Google Chrome 浏览器中显示该页面时,类似信号干扰的屏幕文字动画如图 293-1 所示。有关此实例的主要代码如下。



图 293-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  body { background: black;font-family: 'Varela', sans-serif;}
  .glitch { color: white; font-size:50px; position: relative;
           width: 450px; margin: 0 auto; }
@keyframes myNoise { 0% { clip: rect(22px, 9999px, 94px, 0); }
5% { clip: rect(18px, 9999px, 97px, 0); }10% { clip: rect(57px, 9999px, 27px, 0); }
15% { clip: rect(2px, 9999px, 74px, 0); }20% { clip: rect(26px, 9999px, 55px, 0); }
25% { clip: rect(22px, 9999px, 19px, 0);} 30% { clip: rect(86px, 9999px, 47px, 0); }
35% { clip: rect(54px, 9999px, 45px, 0);}40% { clip: rect(63px, 9999px, 61px, 0); }
45% { clip: rect(54px, 9999px, 19px, 0);}50% { clip: rect(10px, 9999px, 27px, 0); }
55% { clip: rect(38px, 9999px, 48px, 0);} 60% { clip: rect(49px, 9999px, 50px, 0); }
65% { clip: rect(84px, 9999px, 69px, 0);} 70% { clip: rect(66px, 9999px, 62px, 0); }
75% { clip: rect(18px, 9999px, 45px, 0);}80% { clip: rect(65px, 9999px, 54px, 0); }
85% { clip: rect(11px, 9999px, 16px, 0);} 90% { clip: rect(74px, 9999px, 33px, 0); }
95% { clip: rect(88px, 9999px, 75px,0);}100% {clip: rect(35px, 9999px, 28px, 0); } }
.glitch:after { content: attr(data-text); position: absolute; left: 2px;
      top: 0; color: white; background: black; overflow: hidden;
      animation: myNoise 2s infinite linear alternate-reverse;}
 </style></head>
<body><center><div class="glitch" data-text="测试信号干扰花屏效果">测试信号干扰花屏效果</div>
</center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation: myNoise 2s infinite linear alternate-reverse 表示在两秒内以 linear 模式执行 myNoise 定义的关键帧动画，并且永不停歇。alternate-reverse 表示动画在奇数次(1、3、5…)时反向播放，在偶数次(2、4、6…)时正向播放。clip 属性用于定义一个剪裁矩形，对于一个绝对定位元素，在这个矩形内的内容才可见，超出这个区域的内容会根据 overflow 的值来处理，剪裁区域可能比元素的内容区大，也可能比内容区小。

此实例的源文件名是 myHtmlB221.html。

294 使用关键帧动画模拟理发店跑马灯特效

此实例主要在关键帧动画中不断改变透明背景图像的水平位置并使用背景图像填充文字线条，从而使文字线条出现理发店跑马灯的滚动变换特效。当在 Google Chrome 浏览器中显示该页面时，随着背景图像水平位置的改变，由透明和不透明间隔构成的背景图像不断交错填充文字线条，因此文字线条也交错呈现透明和不透明的跑马灯的滚动变换特效，如图 294-1 所示。有关此实例的主要代码如下。



图 294-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  p { display: inline-block;width:40em; }
  p span { font: 700 4em/1 "Oswald", sans-serif; background: url(img/B225.png);
          -webkit-background-clip: text; -webkit-text-fill-color: transparent;
          -webkit-animation: myPos 80s linear infinite; }
  @-webkit-keyframes myPos { 0% { background-position: 0% 50%; }
                             100% { background-position: 100% 50%; } }
 </style></head>
<body><p><span>问君能有几多愁</span></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-animation：myPos 80s linear infinite 表示在 80 秒内以 linear 模式执行 myPos 定义的关键帧动画(即从头到尾改变背景图像的水平位置)，且永不停歇；background-position：0% 50%用于设置背景图像的起始位置，0%是水平位置、50%是垂直位置，左上角表示为 0% 0%、右下角表示为 100% 100%，如果仅规定了一个值，另一个值将是 50%；-webkit-background- clip：text 表示背景图像的哪些地方可以显示，如果同时设置了 -webkit-text-fill-color：transparent，则在背景图像中与文字线条对应的部分可以显示。

此实例的源文件名是 myHtmlB225. html。

295 使用关键帧动画高仿加载间隔转圈特效

此实例主要在关键帧动画中不断改变圆环的颜色和角度，从而实现类似于谷歌浏览器加载网页时的转圈动画特效。当在 Google Chrome 浏览器中显示该页面时，隔断的圆环将沿着顺时针方向不断转圈，如图 295-1 所示。有关此实例的主要代码如下。

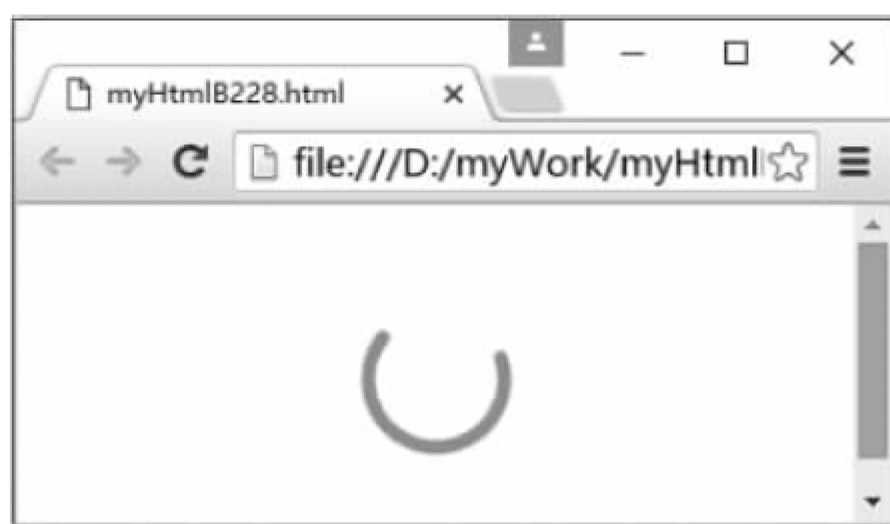


图 295-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   html, body { height: 100%; }
   /* 居中设置 */
   body {display: -webkit-box; display: -webkit-flex; -webkit-align-items: center;
        -webkit-box-pack: center; -webkit-justify-content: center;}
   /* 执行 SVG 关键帧动画 */
   .spinner { -webkit-animation: rotator 1.4s linear infinite; }
   @-webkit-keyframes rotator { 0% { -webkit-transform: rotate(0deg); }
                              100% { -webkit-transform: rotate(270deg); }}
  .path { stroke-dasharray: 187; stroke-dashoffset: 0;
          -webkit-transform-origin: center; transform-origin: center;
```

```
        /* 执行两个关键帧动画 */
        -webkit-animation: dash 1.4s ease-in-out infinite, colors 5.6s ease-in-out infinite; }
   /* 颜色关键帧 */
   @-webkit-keyframes colors { 0% { stroke: #4285F4; } 25% { stroke: #DE3E35; }
     50% { stroke: #F7C223; } 75% { stroke: #1B9A59; } 100% { stroke: #4285F4; } }
    /* 圆环关键帧 */
    @-webkit-keyframes dash { 0% { stroke-dashoffset: 187; }
      50% { stroke-dashoffset: 46.75; -webkit-transform: rotate(135deg); }
      100% { stroke-dashoffset: 187; -webkit-transform: rotate(450deg); } }
 </style></head>
<body><center><svg class="spinner" width="65px" height="65px" viewBox="0 0 66 66"><circle class=
"path" fill="none" stroke-width="6" stroke-linecap="round" cx="33" cy="33" r="30"></circle>
</svg></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-animation: rotator 1.4s linear infinite 用于改变 SVG 的角度;-webkit-animation: dash 1.4s ease-in-out infinite, colors 5.6s ease-in-out infinite 包含两个关键帧动画,第一个关键帧动画用于旋转隔断圆环 stroke 的角度,第二个关键帧动画用于改变隔断圆环 stroke 的颜色;stroke-dasharray:187 用于创建虚线;stroke-dashoffset:187 指定了 dash 到路径开始的距离;-webkit-transform: rotate(135deg)表示顺时针旋转 135°。

此实例的源文件名是 myHtmlB228.html。

296 使用关键帧动画同时旋转多个 3D 汉字

此实例主要通过在关键帧动画中使用 rotateY()方法实现同时旋转多个 3D 汉字。当在 Google Chrome 浏览器中显示该页面时,“何当共剪西窗烛”中的每个字即围绕自身的中心在垂直方向(Y 轴)上不停旋转,旋转成反面的 3D 汉字如图 296-1 所示。有关此实例的主要代码如下。



图 296-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script>
  //为单行中的每个字添加<span>标签
  $(function() {
   word=$('p').html();
   word=word.split('');
   var str=[]; var str_html="";
   $.each(word, function(i, item) {
     str[i]="<span>"+item+"</span>";
   });
   $.each(str, function (i, item) {
```

```
        str_html += item;
      });
      $('p').html(str_html);
  });
  </script>
  <style>
    body {/* 设置灰色背景 */ background - color: gray; }
    /* 为<span>标签中的内容添加旋转特效 */
    p span { display: inline - block; color: black; letter - spacing: 10px;
             /* 创建阴影文字 */
             text - shadow:  - 1px  - 1px 0 #FFF, 1px 1px 0 #333, 1px 1px 0 #444;
             font: bold 60px/100% "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans;
             - webkit - animation: myRotate 3s ease - in - out infinite; }
    @ - webkit - keyframes myRotate { 0% { transform: rotateY(0deg); }
      75% { transform: rotateY(180deg); } 100% { transform: rotateY(360deg); } }
  </style></head>
<body><center><p>何当共剪西窗烛</p></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-animation：myRotate 3s ease-in-out infinite 表示在 3 秒内以 ease-in-out 模式执行 myRotate 关键帧中的 rotateY 旋转动作，且永不停歇；rotateY(180deg)表示元素(汉字)以 3D 方式围绕 Y 轴旋转 180°；letter-spacing：10px 表示每个汉字之间的间隔是 10px。在 CSS 中，letter-spacing 属性用于增加或减少字符间的空白(字符间距)，该属性定义了在文本字符框之间插入多少空间。由于字符字形通常比其字符框要窄，在指定长度值时会调整字母之间通常的间隔，因此其属性值为 normal 就相当于值为 0。

此实例的源文件名是 myHtmlB232.html。

297　使用关键帧动画实现表格隔行闪烁显示

此实例主要通过使用 table tr：nth-child(2n+1)和 table tr：nth-child(2n)选择器筛选表格的所有奇数行和偶数行，并在关键帧动画中设置 background-color 属性，从而实现表格错行闪烁的特效。当在 Google Chrome 浏览器中显示该页面时，浅绿色的背景将交替在表格的奇数行和偶数行闪烁，如图 297-1 所示。有关此实例的主要代码如下。

罗博克葡萄酒	西班牙	歌海娜
科洛文红葡萄酒	法国	丹魄
马蒂尔伯爵葡萄酒	意大利	朗布罗斯科
帝富干红葡萄酒	西班牙	胡米亚
智象美露干红葡萄酒	智利	中央山谷
卡梅罗西干红葡萄酒	法国	赤霞珠

图　297-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
  <style type = "text/css">
    table, tr, td { border: 1px solid gray; text - align: center; }
    /* 奇数行动画 */
    table tr:nth - child(2n + 1) { - webkit - animation: myOdd 1.2s ease infinite; }
```

```
/*偶数行动画*/
table tr:nth-child(2n) { -webkit-animation: myEven 1.2s ease infinite;}
/*奇数行关键帧*/
@-webkit-keyframes myOdd { 0% { background-color: initial; }
  50% { background-color: lightgreen; } 100% { background-color: initial; } }
/*偶数行关键帧 */
 @-webkit-keyframes myEven { 0% { background-color: lightgreen; }
  50% { background-color: initial; } 100% { background-color: lightgreen; } }
 table { margin: 0 auto; width: 400px; border-collapse: collapse;
         text-align: center; font-size: small; border: 1px; }
</style></head>
<body><center><div style="width:400px"><table id="mytable">
    <tr><td>罗博克葡萄酒</td><td>西班牙</td><td>歌海娜</td></tr>
    <tr><td>科洛文红葡萄酒</td><td>法国</td><td>丹魄</td></tr>
    <tr><td>马蒂尔伯爵葡萄酒</td><td>意大利</td><td>朗布罗斯科</td></tr>
    <tr><td>帝富干红葡萄酒</td><td>西班牙</td><td>胡米亚</td></tr>
    <tr><td>智象美露干红葡萄酒</td><td>智利</td><td>中央山谷</td></tr>
    <tr><td>卡梅罗西干红葡萄酒</td><td>法国</td><td>赤霞珠</td></tr></table></div></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，table tr:nth-child(2n+1) { -webkit-animation: myOdd 1.2s ease infinite; }表示表格中的所有奇数行在1.2秒内以ease模式执行myOdd关键帧中的改变背景动作，且永不停歇；table tr:nth-child(2n) {-webkit-animation: myEven 1.2s ease infinite;}表示表格中的所有偶数行在1.2秒内以ease模式执行myEven关键帧中的改变背景动作，且永不停歇；background-color: initial表示以默认值设置背景色；background-color: lightgreen表示以浅绿色设置背景色。需要注意的是，@-webkit-keyframes myOdd { }和@-webkit-keyframes myEven { }两个关键帧的每个点位（百分比）上的动作总是不同（错位），因此在运行时才产生那种交替闪烁的效果。

此实例的源文件名是myHtmlB237.html。

298 使用关键帧动画实现文本框中的提示的闪烁显示

此实例主要通过使用关键闪烁显示帧动画设置文本框的提示文本的color属性，从而实现文本框中的提示内容的闪烁显示特效。当在Google Chrome浏览器中显示该页面时，如果“专业名称：”文本框失去焦点（即能够显示提示文本），则该文本框中的提示文本“教育部规范的专业名称”将以蓝、红、绿交替闪烁；如果“毕业论文：”文本框失去焦点（即能够显示提示文本），则该文本框中的提示文本“公开发表的论文标题”将以红、黄、紫交替闪烁，如图298-1所示。有关此实例的主要代码如下。

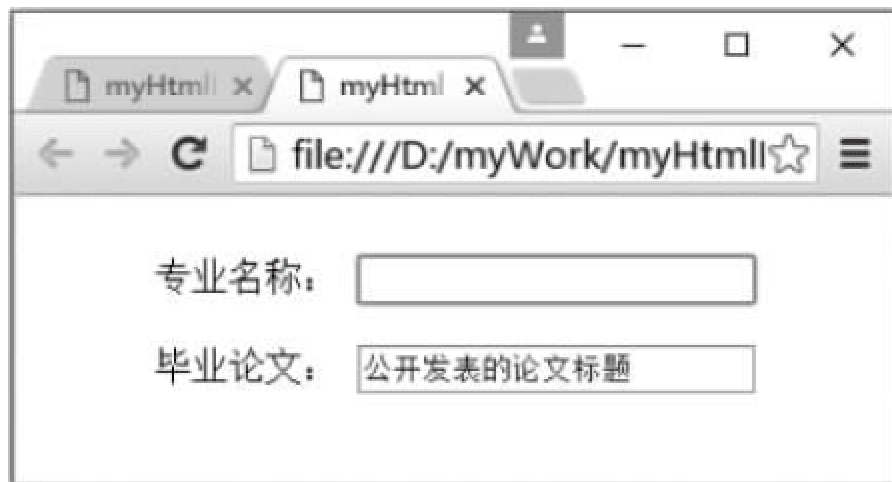


图 298-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $('#myMajor').val()=="教育部规范的专业名称" ? $('#myMajor').toggleClass('majorBlink') : 0;
   //myMajor获得焦点时响应,即移除提示
   $('#myMajor').focus(function(){
    if($(this).val()=="教育部规范的专业名称"){
      $(this).val("");
     //如果动画类majorBlink存在,则移除之
      $(this).hasClass('majorBlink') ? $(this).removeClass('majorBlink') : 0;
   }});
   //myMajor失去焦点时响应,即添加提示
   $('#myMajor').blur(function(){
    if ($(this).val().length==0) {
     $(this).val("教育部规范的专业名称");
     //toggleClass()对设置或移除被选元素的一个或多个类进行切换,
     //即添加动画类majorBlink
     $(this).toggleClass('majorBlink');
   }});
   $('#myPaper').val()=="公开发表的论文标题" ? $('#myPaper').toggleClass('paperBlink') : 0;
   //myPaper获得焦点时响应,即移除提示
   $('#myPaper').focus(function(){
    if($(this).val()=="公开发表的论文标题"){
      $(this).val("");
     //如果动画类paperBlink存在,则移除之
      $(this).hasClass('paperBlink') ? $(this).removeClass('paperBlink') : 0;
   }});
   //myPaper失去焦点时响应,即添加提示
   $('#myPaper').blur(function(){
    if ($(this).val().length==0) {
     $(this).val("公开发表的论文标题");
      //添加动画类paperBlink
     $(this).toggleClass('paperBlink');
 }});});
 </script>
 <style type="text/css">
  .majorBlink{-webkit-animation:majorBlink ease 1s infinite alternate;}
   @-webkit-keyframes majorBlink{ 0% {color:blue;} 50% {color:red;}
                                 100% {color:lightgreen;} }
  .paperBlink{-webkit-animation:paperBlink ease 1s infinite alternate;}
   @-webkit-keyframes paperBlink{ 0% {color:red;} 50% {color: yellow;}
                                 100% {color:purple;} }
 </style></head>
<body><center><br>
 <section>专业名称:<input value="教育部规范的专业名称" id="myMajor"/></section><br>
 <section>毕业论文:<input value="公开发表的论文标题" id="myPaper"/></section><br></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.majorBlink{-webkit- animation: majorBlink ease 1s infinite alternate;}表示在majorBlink文本框中的提示文本将在1秒内以ease模式执行majorBlink关键帧中的改变文本颜色动作,每次执行后均反向再执行(alternate),且永不停歇;50% {color:red;}表示在0.5秒时设置提示文本的颜色为红色。

此实例的源文件名是 myHtmlB238. html。

299 使用关键帧动画来回水平扫描阴影文本

此实例主要通过使用关键帧动画改变背景图像的水平位置,从而实现来回水平扫描阴影文本的特殊效果。当在 Google Chrome 浏览器中显示该页面时,渐变的背景将从左向右再从右向左来回滑动,从而使文字依次呈现半隐半显的效果,如图 299-1 所示。有关此实例的主要代码如下。

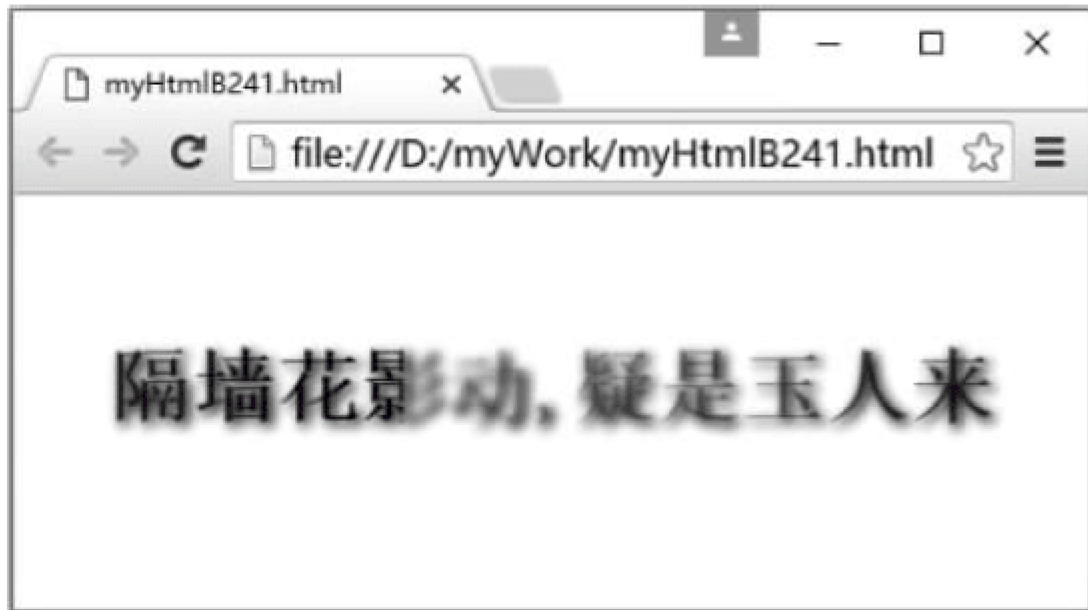


图 299-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { width: 400px; height:150px; line-height: 150px; font-weight: bold;
        font-size: 36px; text-shadow:2px 2px 6px #000000;
        /* 创建向右的渐变背景 */
        background: -webkit-linear-gradient(right, black, white);
        /* 使用背景色设置文本颜色 */
        -webkit-background-clip: text; -webkit-text-fill-color: transparent;
        -webkit-animation: myScan 5s ease infinite alternate; }
   @-webkit-keyframes myScan { 0% { background-position-x: 0px; }
                               100% { background-position-x: 400px; } }
 </style></head>
<body><center><div>隔墙花影动,疑是玉人来</div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-position-x: 400px 表示背景图像的水平位置在 400px;-webkit-animation: myScan 5s ease infinite alternate 表示线性渐变背景图像将在 5 秒内以 ease 模式执行 myScan 关键帧中的改变水平位置动作,每次执行后均反向再执行(alternate),且永不停歇。

此实例的源文件名是 myHtmlB241. html。

300 使用关键帧动画动态拉伸超链接下画线

此实例主要使用关键帧动画改变 div 元素(即超链接下画线)的宽度,从而实现在鼠标指针悬浮于超链接时动态拉伸(扩展)下画线的宽度的效果。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在"中国人民大学"超链接上,则红色的下画线将从中心向两端拉伸(扩展),如图 300-1 所示。当然,如果鼠标指针悬浮在"清华大学"超链接上,也将实现类似的效果。有关此实例的主要代码如下。



图 300-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $('#myTHA').mouseenter(function() {//鼠标进入时响应
    //先显示下画线,再扩展宽度
    $('#myTH').addClass('hide') ? $('#myTH').removeClass('hide'). addClass('scale') :
$('#myTH').addClass('scale');
   });
   $('#myTHA').mouseleave(function() {//鼠标指针离开时响应
    //先停止扩展动作,再隐藏下画线
    $('#myTH').hasClass('scale') ? $('#myTH').removeClass('scale'). addClass('hide') :
$('#myTH').addClass('hide');
   });
   $('#myPKA').mouseenter(function() {//鼠标指针进入时响应
    //先显示下画线,再扩展宽度
    $('#myPK').addClass('hide') ? $('#myPK').removeClass('hide'). addClass('scale') :
$('#myPK').addClass('scale');
   });
   $('#myPKA').mouseleave(function() {//鼠标指针离开时响应
    //先停止扩展动作,再隐藏下画线
    $('#myPK').hasClass('scale') ? $('#myPK').removeClass('scale'). addClass('hide') :
$('#myPK').addClass('hide');
  });});
 </script>
 <style type="text/css">
   * { margin: 0px; padding: 0px;}
   .myTH { width: 70px; height: 2px; background-color: red;}
   .myPK { width: 95px; height: 2px; background-color: red;}
   .scale { /* 将下画线的宽度从 0px 扩展到原始宽度 */
            -webkit-animation: myScale 1s ease; }
   .hide { /* 隐藏下画线 */ opacity: 0; }
   @-webkit-keyframes myScale { 0% { -webkit-transform: scaleX(0);}
                                100% { -webkit-transform: scaleX(1); }}
```

```
    a { /* 去掉超链接默认的下画线 */ text-decoration: none; }
    .myBox { padding-top: 20px; }
    .myDiv { width: 400px; height: 250px; border-radius: 5px;
             background-image: url(img/B247.jpg); margin-top: 5px; }
  </style></head>
 <body><center><div class="myDiv">
  <p class="myBox"><a id="myTHA" href="http://www.tsinghua.edu.cn">清华大学</a>
  <div id="myTH" class="hide myTH"></div></p>
 <p class="myBox"><a id="myPKA" href="http://www.ruc.edu.cn">中国人民大学</a>
  <div id="myPK" class="hide myPK"></div></p></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：scaleX(0)表示将下画线的宽度缩减到无；-webkit-transform：scaleX(1)表示将下画线的宽度恢复到原始宽度；-webkit-animation：myScale 1s ease 表示在 1 秒内按照 ease 模式执行 myScale 关键帧中的宽度拉伸动作。

此实例的源文件名是 myHtmlB247.html。

301　使用关键帧动画实现打字式的输入效果

此实例主要使用关键帧动画改变字符串的 width 属性，并在 animation 属性中使用 steps，从而实现打字式的输入效果。当在 Google Chrome 浏览器中显示该页面时，“Bad news has wings”将从左到右逐个字母显示出来，然后再从右到左逐个字母清除掉，如此反复循环地演示打字效果，如图 301-1 所示。有关此实例的主要代码如下。



图　301-1

```
 <!doctype html><html><head><meta charset="UTF-8">
  <style type="text/css">
   * { margin: 0; padding: 0; }
   /* 设置背景 */
   div { text-align: center; margin: 10px auto; width: 400px; height: 250px;
        border-radius: 10px; -webkit-box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.3);
        position: relative; background-image: url(img/B250.jpg);
        background-size: 100% 100%; }
   /* 演示打字效果 */
```

```
    h1 { font: bold 200 % Consolas, Monaco, monospace; border - right: 0.1em solid;
         / * 字符串(Bad news has wings)的长度,字母和空格的个数 * /
         width: 19ch; margin: 2em 1em; white - space: nowrap; overflow: hidden;
         / * steps(19)一定要等于字符串的长度 19 * /
        animation: typing 10s steps(19) infinite alternate;
        color: aqua; position: absolute; top:30px; }
   @keyframes typing { from { width: 0px; } }
 </style></head>
<body><div><h1>Bad news has wings</h1></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation: typing 10s steps(19) infinite alternate 表示在 10 秒内按照 steps(19)跳跃模式执行 typing 关键帧中的动作，然后再逆向执行，永不停歇。在 CSS3 中，animation 默认以 ease 方式过渡，它会在每个关键帧之间插入补间动画，所以动画效果是连贯性的。除了 ease、linear、cubic-bezier 之类的过渡函数都会为其插入补间。但有些效果不需要补间，只需要关键帧之间的跳跃，这时应该使用 steps 过渡方式。在此实例中，steps(19)表示让整个动画在 19 个关键帧之间切换，因为"Bad news has wings"包含了 19 帧(字母和空格)，所以这里设置了 19。

此实例的源文件名是 myHtmlB250.html。

302　使用关键帧动画高仿足球滚动效果

此实例主要通过在关键帧动画中同时执行旋转和平移两个动作高仿足球滚动的动态效果。当在 Google Chrome 浏览器中显示该页面时，足球将从草坪的左端滚动到右端，然后再从右端滚动到左端，如此循环，永不停歇，如图 302-1 所示。有关此实例的主要代码如下。

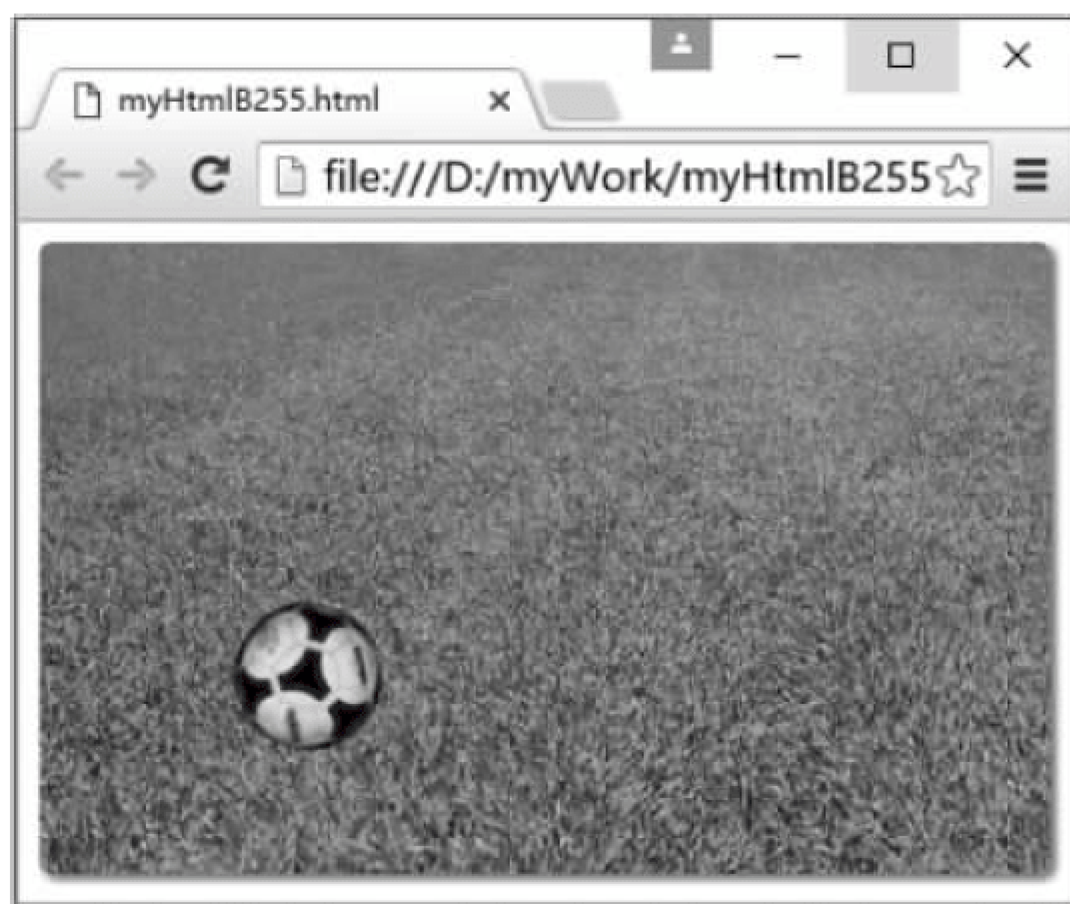


图　302-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  / * 设置草坪样式 * /
  .myBox { width: 400px; height: 250px; background - image: url("img/B255B.jpg");
     background - size: 100 % 100 % ; border - radius: 5px;box - shadow: 2px 2px 5px gray; }
  / * 设置足球滚动 * /
```

```
.myBall { position: absolute; left:15px; top:150px;
          width: 60px; height: 60px; background-size: 100% 100%;
          border-radius: 50%; background-repeat: no-repeat;
          background-image:url(img/B255.png);
          -webkit-animation: myRoll 5s infinite alternate; }
@-webkit-keyframes myRoll {
          0% { -webkit-transform: translateX(0px) rotateZ(0deg); }
          100% { -webkit-transform: translateX(330px) rotateZ(360deg);} }
</style></head>
<body><div class="myBox"><div class="myBall"></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，translateX(0px)和 translateX(330px)表示足球(div)将从 0px 位置沿着 X 轴平移到 330px 位置；rotateZ(0deg)和 rotateZ(360deg)表示足球将围绕 Z 轴从 0°旋转至 360°，即一圈；-webkit-animation：myRoll 5s infinite alternate 表示在 5 秒内完成关键帧动画 myRoll 中的旋转和平移动作，然后再逆向执行，永不停歇。

此实例的源文件名是 myHtmlB255.html。

303 使用关键帧动画实现逐帧播放特效

此实例主要通过在 animation 属性中指定 steps()函数实现卡通动画的逐帧播放效果。当在 Google Chrome 浏览器中显示该页面时将动态显示美女的走路效果，如图 303-1 所示。有关此实例的主要代码如下。

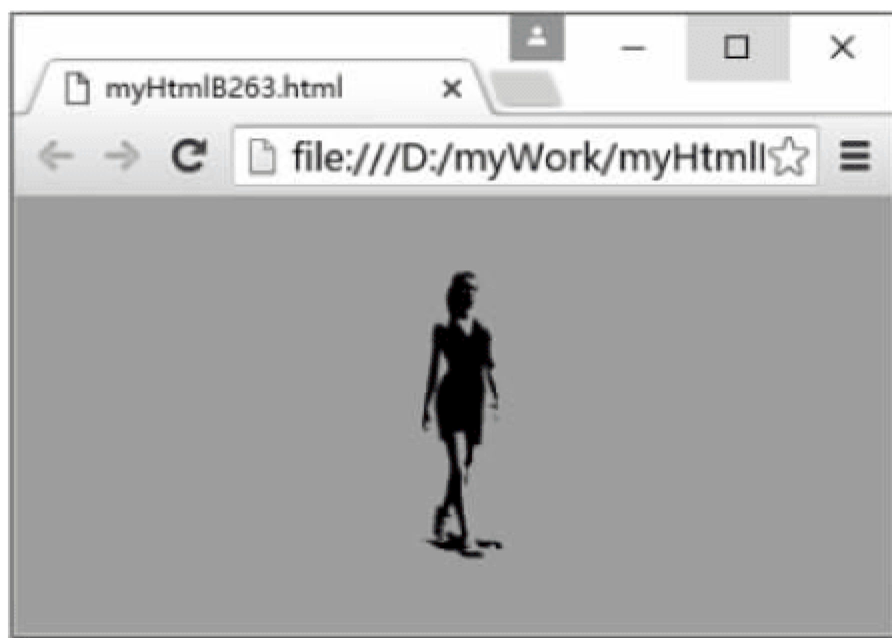


图 303-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  @-webkit-keyframes myBeauty { 0% { background-position: -0% 0px; }
                                /*原图像的宽度是 378 = 54×7*/
                                100% { background-position: -700% 0px; } }
  div { margin: 20px auto; height: 150px;width: 54px;
        /*steps(7)表示让整个动画在 7 个关键帧之间切换*/
        -webkit-animation: myBeauty 700ms steps(7) infinite;
        background: url(img/B263.png); }
  body { background-color: aqua; }
 </style></head>
<body><div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-animation：myBeauty 700ms

steps(7) infinite 表示在 700 毫秒内完成关键帧动画 myBeauty 指定的改变背景图像水平坐标的动作，并且分成 7 步来执行，永不停歇。在 CSS3 中，animation 属性指定的动画默认以 ease 方式过渡，它会在每个关键帧之间插入补间动画，所以动画效果是连贯性的。但是有些动画效果并不需要补间，只需要关键帧之间的跳跃，这时就应该使用 steps 过渡方式。steps()函数指定了一个阶跃函数，第一个参数指定了时间函数中的间隔数量(必须是正整数)，第二个参数可选，接受 start 和 end 两个值，指定在每个间隔的起点或是终点发生阶跃变化，默认为 end。step-start 等同于 steps(1，start)，动画分成一步，动画执行时以左侧端点的部分为开始。step-end 等同于 steps(1，end)，动画分成一步，动画执行时以结尾端点为开始，默认值为 end。实例中的 steps(7)表示让整个动画在 7 个关键帧之间切换。由于原图像中包含了 7 个动作，如图 303-2 所示，所以这里设置了 7，而且动画时长是 700ms，也就是说每一帧停留 100ms，这就和普通的 GIF 动图达到了一样的效果。



图 303-2

background-position 属性用于设置或检索元素的背景图像位置，在使用此属性时必须先指定元素的 background-image 属性。background-position 属性提供两个参数值，第一个用于横坐标，第二个用于纵坐标，如果只提供一个，该值将用于横坐标，纵坐标将默认为 50%。其默认值是 0% 0%，效果等同于 left top。background-position 属性的语法格式如下。

```
background-position:<position>[ , <position> ]*
<position>=[ left | center | right | top | bottom | <percentage> | <length> ] | [ left | center | right
| <percentage> | <length> ] [ top | center | bottom | <percentage> | <length> ] | [ center | [ left |
right ] [ <percentage> | <length> ]? ] && [ center | [ top | bottom ] [ <percentage> | <length> ]? ]
```

各选项的相关说明如下。

(1) <percentage>：表示用百分比指定背景图像填充的位置，可以为负值。

(2) <length>：表示用长度值指定背景图像填充的位置，可以为负值。

(3) center：表示背景图像横向和纵向居中。

(4) left：表示背景图像在横向上填充从左边开始。

(5) right：表示背景图像在横向上填充从右边开始。

(6) top：表示背景图像在纵向上填充从顶部开始。

(7) bottom：表示背景图像在纵向上填充从底部开始。

此实例的源文件名是 myHtmlB263.html。

304 使用关键帧动画高仿扑克出牌前的弹跳动作

此实例主要在 animation 属性中调用改变图像纵坐标的动画，从而实现高仿扑克出牌前的似出非出的(抽插)弹跳动作。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针放在第二张扑

克牌上，则该扑克牌将上下弹跳，似出非出，如图 304-1 所示；如果将鼠标指针放在其他扑克牌上，仍将实现类似的效果。有关此实例的主要代码如下。



图　304-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  * { margin: 0; padding: 0;}
  /* 设置扑克牌样式 */
  img{ border - radius: 10px;float: left;height: 150px; width: 100px;
       margin: 50px auto;}
  /* 设置鼠标指针悬浮于扑克牌时的弹跳动画 */
  img:hover { position: relative; - webkit - animation: myBounce 0.5s ease infinite; }
  @ - webkit - keyframes myBounce { 0 % { top: - 35px; } 100 % { top: 35px; } }
 </style></head>
<body><div><span><img src = "img/B265A. jpg"></span>
 <span><img src = "img/B265B. jpg"></span><span><img src = "img/B265C. jpg"></span>
 <span><img src = "img/B265D. jpg"></span></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，0% {top: －35px;}表示在弹跳动画开始时扑克牌的纵坐标为 top: －35px，100% {top: 35px;}表示在弹跳动画结束时扑克牌的纵坐标为 top: 35px，至于扑克牌的纵坐标在动画期间怎样从－35px 移动到 35px，ease 会自动采用补间策略解决这一问题；-webkit-animation: myBounce 0.5s ease infinite 表示在 500 毫秒内按照 ease 模式完成关键帧动画 myBounce 指定的改变扑克牌纵坐标的动作，且永不停歇。

此实例的源文件名是 myHtmlB265.html。

305　使用关键帧动画往返滑动自定义下画线

此实例主要在 animation 属性中调用改变背景的水平坐标的动画，从而实现自定义下画线的往返滑动特效。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“《巴山夜雨》、《天云山传奇》”上，则将在下面出现一条类似于虚线的下画线，并且往返滑动，悬浮期间永不停歇，如图 305-1 所示；如果将鼠标指针悬浮在其他电影片名上，仍将实现类似的效果。有关此实例的主要代码如下。



图 305-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /*绘制阴影框*/
  div { display: inline-block;width: 380px; height: 200px; padding-left: 20px;
       padding-right: 20px; padding-bottom: 20px;margin: 10px; border-radius: 5px;
       background-color: #FFF;border: 1px solid #EEE; position: relative;
       box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; }
  body { background-color: lightgray;}
  /*设置字体大小,必须与 span 中的字体大小一致*/
  p { line-height: 35px; font-family: 楷体; font-size: 16px; }
  span { position: absolute; height: 35px; font-size: 16px; }
  /*定制下画线的风格*/
  span:hover{ background: linear-gradient(90deg,red 90%, transparent 0) repeat-x;
   background-size: 8px 2px;
   /*设置动画轮流反向播放并且永不停止*/
    -webkit-animation: mySlide 1s ease alternate infinite; }
   /*改变下画线背景的水平坐标位置*/
   @-webkit-keyframes mySlide { 0% { background-position: -3em 30px; }
                              100% { background-position: 0 30px; }}
 </style></head>
<body><div><p>最佳故事片:<span>«巴山夜雨»､«天云山传奇»</span>«巴山夜雨»､«天云山传奇»。<br>
    最佳科教片:<span>«生命与蛋白质-人工合成胰岛素»</span>«生命与蛋白质-人工合成胰岛素»。
<br>
    最佳纪录片:<span>«蛇口奏鸣曲»</span>«蛇口奏鸣曲»。<br>
    最佳儿童片:<span>«烛光里的微笑»</span>«烛光里的微笑»。<br>
    最佳合拍故事片:<span>«宋氏三姐妹»</span>«宋氏三姐妹»。<br></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-position：0 30px 表示自定义下画线(背景)的水平坐标为 0、垂直坐标为 30px，background-position：－3em 30px 表示自定义下画线(背景)的水平坐标为－3em、垂直坐标为 30px(在一般情况下，如果元素的坐标值为负数，它们就是不可见的)；-webkit-animation：mySlide 1s ease alternate infinite 表示在 1000 毫秒内按照 ease 模式完成关键帧动画 mySlide 指定的改变下画线水平坐标(从－3em 到 0)的动作，并且往返滑动，永不停歇。

此实例的源文件名是 myHtmlB276.html。

306 使用关键帧动画实现白云和阴影的联动

此实例主要通过控制两个动画相同的持续时间实现白云在飘移时和对应阴影的联动。当在 Google Chrome 浏览器中显示该页面时，如果天上的白云向上飘移，则下面的与白云对应的椭圆形阴影随之变小，如图 306-1 所示；如果天上的白云向下飘移，则下面的与白云对应的椭圆形阴影随之变大。有关此实例的主要代码如下。



图 306-1

```
<!doctype html><html><head><meta charset="UTF-8">
  <style type="text/css">
   .myCloudy{ position: absolute; top: 50%; left: 50%;
            width: 400px; height: 250px;transform: translate(-50%, -50%);
            background:url(img/B281.jpg); border-radius: 5px; }
   .myCloudy:before { content: ""; position: absolute; height: 50px;
     width: 50px; border-radius: 50%; background: white;left: 20%;top: 30%;
     transform: translate(-50%, -40%);
     /*绘制整团白云*/
     box-shadow: white 65px -15px 0 -5px, white 25px -25px, white 30px 10px,
                white 60px 15px 0 -10px, white 85px 5px 0 -5px, lightcyan 35px -35px,
                lightcyan 66px -27px 0 -1px, lightcyan 91px -10px 0 -2px;
    /*白云动画*/ animation: myCloudy 5s ease-in-out infinite; }
   .myCloudy:after { content: ""; position: absolute; top: 80%; left: 50%;
    height: 15px; width: 120px; background: rgba(0, 0, 0, 0.5);border-radius: 50%;
    transform: translate(-50%, -50%);
    animation: myCloudy_shadow 5s ease-in-out infinite; }
   /*白云关键帧 */
   @keyframes myCloudy { 50% { transform: translate(-50%, -80%); }
                       100% { transform: translate(-50%, -40%); } }
   /*阴影关键帧*/
   @keyframes myCloudy_shadow {50% { transform: translate(-50%, -50%) scale(0.8); }
                             100% { transform: translate(-50%, -50%) scale(1); } }
 </style></head>
<body><div class="myCloudy"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform: translate(−50%, −80%)

表示将元素(白云)平移到参数中指定的位置；animation：myCloudy 5s ease-in-out infinite 表示在 5 秒内按照 ease-in-out 模式完成关键帧动画 myCloudy 指定的 translate 改变元素(白云)纵坐标(从−80%到−40%)的动作，且永不停歇；transform：translate(−50%，−50%) scale(0.8)表示将元素(阴影)平移到参数中指定的位置(此实例没有改变，因为前后坐标均完全相同，用户可根据需要考虑此问题)，并缩小 scale 到 80%；animation：myCloudy_shadow 5s ease-in-out infinite 表示在 5 秒内按照 ease-in-out 模式完成关键帧动画 myCloudy_shadow 指定的 scale 改变元素(阴影)大小(从 80%到100%)的动作，且永不停歇。

此实例的源文件名是 myHtmlB281.html。

307 使用关键帧动画实现白云和文本的联动

此实例主要通过控制两个动画具有的持续时间实现白云在飘移时和内嵌独立文本的同时联动。当在 Google Chrome 浏览器中显示该页面时，如果不规则的白云向上飘移，则其里面内嵌的天气预报文本也同步向上飘移，如图 307-1 所示；如果不规则的白云向下飘移，则其里面内嵌的天气预报文本也同步向下飘移。有关此实例的主要代码如下。



图 307-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myCloudy{position: absolute; top: 50%; left: 50%; width: 400px;
           height: 250px;transform: translate(-50%, -50%);
           background:url(img/B250.jpg); border-radius: 5px; }
  .myCloudy:after { content: ""; position: absolute; height:60px; width: 60px;
                   background: white; left: 20%; top: 30%; border-radius: 50%;
                   /*绘制整团白云*/
                   box-shadow: white 65px -20px 0 -5px, white 25px -25px,
                              white 23px 20px, white 80px 15px 0 -5px, white 95px -15px 0 -9px;
                   transform: translate(-50%, -30%); animation: myPos 5s ease-in-out infinite; }
  .myCloudy:before{content:"北部新区 \D\A 2017年4月4日 \D\A 晴转多云 23℃";
                   position: absolute; top: 30%;left: 32%;height: 65px; width: 100px;
                   white-space: pre; text-align: center; font-size: 14px; line-height: 20px;
                   z-index:10;background: white; transform: translate(-50%, -30%);
                   animation: myPos 5s ease-in-out infinite; }
```

```
    @keyframes myPos { 50 % { transform: translate( - 50 % , - 80 % ); }
                       100 % { transform: translate( - 50 % , - 30 % ); } }
  </style></head>
  <body><div class = "myCloudy"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform: translate (－50%，－80%)表示将元素(白云或内嵌文本)平移到参数中指定的位置；animation: myPos 5s ease-in-out infinite 表示在 5 秒内按照 ease-in-out 模式完成关键帧动画 myPos 指定的 translate 改变元素(白云或内嵌文本)纵坐标(从－80%到－30%)的动作，且永不停歇；white-space: pre 表示空白会被浏览器保留，其行为方式类似 HTML 中的< pre >标签，在此实例中的主要作用是确保 content 属性中的“\D\A”实现换行功能；box-shadow: white 65px －20px 0 －5px，white 25px －25px，white 23px 20px，white 80px 15px 0 －5px，white 95px －15px 0 －9px 用于创建 5 个圆形阴影，当这 5 个圆形阴影叠加在一起的时候就类似于一团不规则的白云。

此实例的源文件名是 myHtmlB282. html。

308　使用关键帧动画实现雨滴从白云中下落

此实例主要在多个关键帧中绘制不同位置的长条形雨滴，然后通过动画改变其位置，从而实现拖尾的雨滴从白云中连续下落的特殊效果。当在 Google Chrome 浏览器中显示该页面时，拖尾的雨滴从白云中下落的效果如图 308-1 所示。有关此实例的主要代码如下。

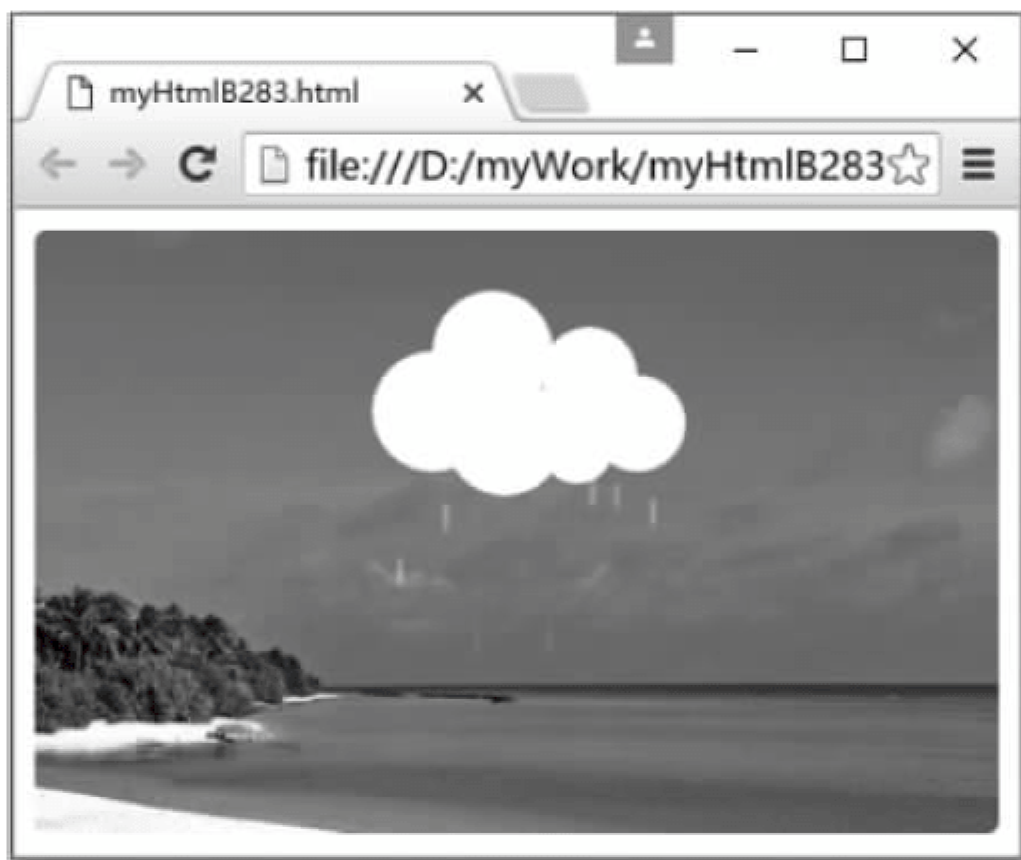


图　308-1

```
  <!doctype html><html><head><meta charset = "UTF - 8">
   <style type = "text/css">
    .myBox {position: absolute; top: 50 % ; left: 50 % ; width: 400px; height: 250px;
           transform: translate( - 50 % , - 50 % );background: url(img/B283. jpg);
           border - radius: 5px;}
     .myRainy { position: absolute; width: 3px; height: 12px; top: 20 % ; left: 50 % ;
                 border - radius: 50 % ; box - shadow: rgba(255,255,255, 0) - 10px 120px, rgba(255,255,
255, 0) 40px 120px, rgba(255,255,255, 0.3) - 50px 75px, rgba(255,255,255, 0.3) 55px 50px, rgba(255,
255,255, 0.3) - 18px 100px, rgba(255,255,255, 0.3) 12px 95px, rgba(255,255,255, 0.3) - 31px 45px, rgba
(255,255,255, 0.3) 30px 35px; animation:myFall 1s infinite linear; }
```

```
.myRainy:before { content: ""; position: absolute;border-radius: 50%;
                  height: 50px; width: 50px;top: 30px; left: -40px;
                  background: white; transform: translate(-40%, -60%);
                  /*绘制整团白云*/
                  box-shadow: white 65px -15px 0 -5px,white 25px -25px, white 30px 10px, white
                  60px 15px 0 -10px, white 85px 5px 0 -5px; }
@keyframes myFall { 0% { box-shadow: rgba(255,255,255, 0) -10px 30px, rgba(255,255,255, 0) 40px
40px, rgba(255,255,255, 0.3) -50px 75px, rgba(255,255,255, 0.3) 55px 50px, rgba(255,255,255, 0.3)
-18px 100px, rgba(255,255,255, 0.3) 12px 95px, rgba(255,255,255, 0.3) -31px 45px, rgba(255,255,255,
0.3) 30px 35px; }
    25% { box-shadow: rgba(255,255,255, 0.3) -10px 45px, rgba(255,255,255, 0.3) 40px 60px, rgba
(255,255,255, 0.3) -50px 90px, rgba(255,255,255, 0.3) 55px 65px, rgba(255,255,255, 0) -18px 120px,
rgba(255,255,255, 0) 12px 120px, rgba(255,255,255, 0.3) -31px 70px, rgba(255,255,255, 0.3) 30px 60px;
}
    26% {box-shadow: rgba(255,255,255, 0.3) -10px 45px, rgba(255,255,255, 0.3) 40px 60px, rgba
(255,255,255, 0.3) -50px 90px, rgba(255,255,255, 0.3) 55px 65px, rgba(255,255,255, 0) -18px 40px,
rgba(255,255,255, 0) 12px 20px, rgba(255,255,255, 0.3) -31px 70px, rgba(255,255,255, 0.3) 30px
60px; }
    50% {box-shadow: rgba(255,255,255, 0.3) -10px 70px, rgba(255,255,255, 0.3) 40px 80px, rgba
(255,255,255, 0) -50px 100px, rgba(255,255,255, 0.3) 55px 80px, rgba(255,255,255, 0.3) -18px 60px,
rgba(255,255,255, 0.3) 12px 45px, rgba(255,255,255, 0.3) -31px 95px, rgba(255,255,255, 0.3) 30px
85px; }
    51% { box-shadow: rgba(255,255,255, 0.3) -10px 70px, rgba(255,255,255, 0.3) 40px 80px, rgba
(255,255,255, 0) -50px 45px, rgba(255,255,255, 0.3) 55px 80px, rgba(255,255,255, 0.3) -18px 60px,
rgba(255,255,255, 0.3) 12px 45px, rgba(255,255,255, 0.3) -31px 95px, rgba(255,255,255, 0.3) 30px
85px; }
     75% { box-shadow: rgba(255,255,255, 0.3) -10px 95px, rgba(255,255,255, 0.3) 40px 100px, rgba
(255,255,255, 0.3) -50px 60px, rgba(255,255,255, 0) 55px 95px, rgba(255,255,255, 0.3) -18px 80px,
rgba(255,255,255, 0.3) 12px 70px, rgba(255,255,255, 0) -31px 120px, rgba(255,255,255, 0) 30px
110px; }
     76% { box-shadow: rgba(255,255,255, 0.3) -10px 95px, rgba(255,255,255, 0.3) 40px 100px, rgba
(255,255,255, 0.3) -50px 60px, rgba(255,255,255, 0) 55px 35px, rgba(255,255,255, 0.3) -18px 80px,
rgba(255,255,255, 0.3) 12px 70px, rgba(255,255,255, 0) -31px 25px, rgba(255,255,255, 0) 30px 15px; }
     100% {box-shadow: rgba(255,255,255, 0) -10px 120px, rgba(255,255,255, 0) 40px 120px, rgba
(255,255,255, 0.3) -50px 75px, rgba(255,255,255, 0.3) 55px 50px, rgba(255,255,255, 0.3) -18px 100px,
rgba(255,255,255, 0.3) 12px 95px, rgba(255,255,255, 0.3) -31px 45px, rgba(255,255,255, 0.3) 30px
35px; }}
 </style></head>
<body><div class="myBox"><div class="myRainy"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：rgba(255, 255, 255, 0) －10px 120px, rgba(255,255,255, 0) 40px 120px, rgba(255,255, 255, 0.3) －50px 75px, rgba(255,255,255, 0.3) 55px 50px, rgba(255,255,255, 0.3) －18px 100px, rgba(255, 255, 255, 0.3) 12px 95px, rgba(255,255,255, 0.3) －31px 45px, rgba(255, 255,255, 0.3) 30px 35px 用于一次性产生多个不同位置的雨滴，它主要是通过设置阴影的不同水平位置和垂直位置来实现的；animation：myFall 1s infinite linear 表示在 1 秒内按照 linear 模式显示在关键帧动画 myFall 中每个百分点指定的雨滴，且永不停歇。

此实例的源文件名是 myHtmlB283.html。

309 使用关键帧动画高仿风力发电机的桨叶转动的特效

此实例主要通过在关键帧动画中旋转 3 个桨叶的角度高仿风力发电机的桨叶转动的动态效果。当在 Google Chrome 浏览器中显示该页面时，风力发电机的桨叶转动的动态效果如图 309-1 所示。有关此实例的主要代码如下。



图 309-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { position: absolute;top: 50%; left: 50%;width: 400px; height: 250px;
          border-radius: 10px;transform: translate(-50%, -50%);
          background: url(img/B284.jpg);}
  /*绘制直杆*/
  .myBox:after { content: ""; position: absolute;top: 68%; left: 50%;
transform: translate(-50%, -50%); width: 6px; height: 120px;background: #FFF; }
  /*绘制第一个桨叶*/
  .myBreeze { position: absolute; top: 30%; left: 50%;
             transform: translate(-50%, -50%) rotate(1deg);
             border-bottom: 60px solid #FFF; border-left: 5px solid transparent;
             border-right: 5px solid transparent;transform-origin: 50.5% 62px;
             animation: myWind 12s infinite linear;}
  /*绘制第二个桨叶*/
  .myBreeze:before { position: absolute; top: 75px;left: -59px;
                    content: "";border-right: 60px solid #FFF;border-top: 5px solid transparent;
                    border-bottom: 5px solid transparent;transform: rotate(-30deg); }
  /*绘制第三个桨叶*/
  .myBreeze:after { position: absolute; top: 75px; left: -1px;
                   content: ""; border-left: 60px solid #FFF;border-top: 5px solid transparent;
                   border-bottom: 5px solid transparent;transform: rotate(30deg); }
  @keyframes myWind { 0% { transform: translate(-50%, -50%) rotate(0deg); }
                     100% { transform: translate(-50%, -50%) rotate(360deg);} }
 </style></head>
<body><div class="myBox"><div class="myBreeze"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，绘制第一(二、三)个桨叶实际上是通过控制边框线宽度来透明(transparent)左、右、底边框线实现的。其中，第一个桨叶的绘制效果如

图 309-2 所示。如果按照下列代码进行修改，即用红色和蓝色代替左边框线和右边框线的透明色，并设置其宽度和高度(以前此值均为 0px)，则可看到第一个桨叶的左、右、底边框线是怎样切分成三角形的，如图 309-3 所示。

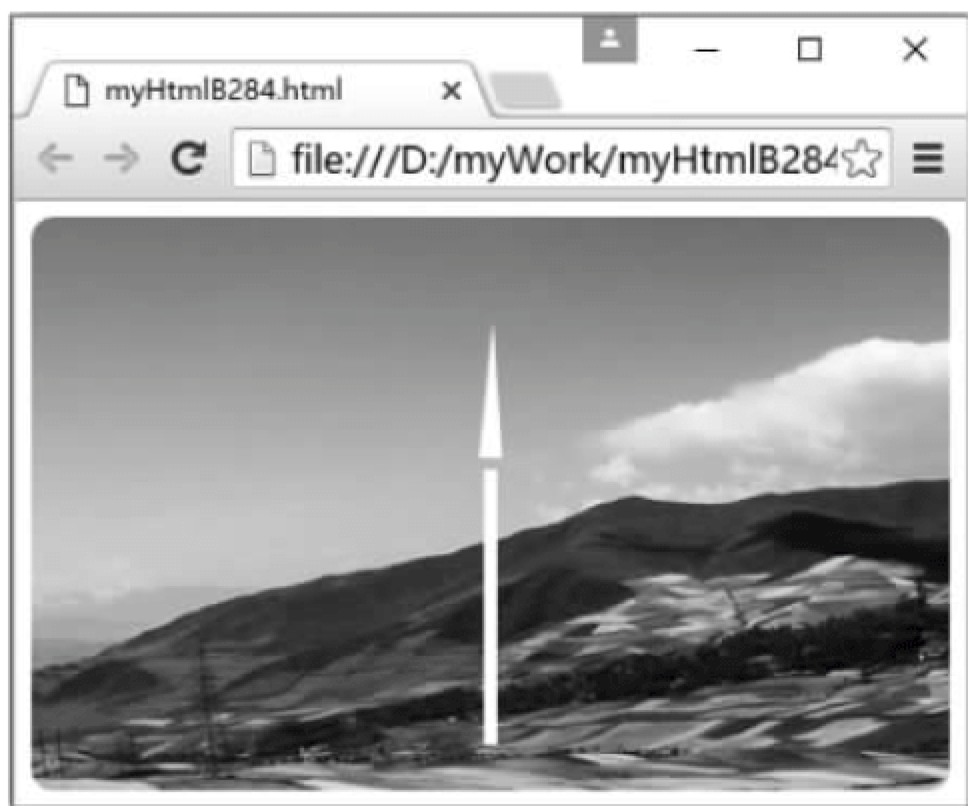


图 309-2

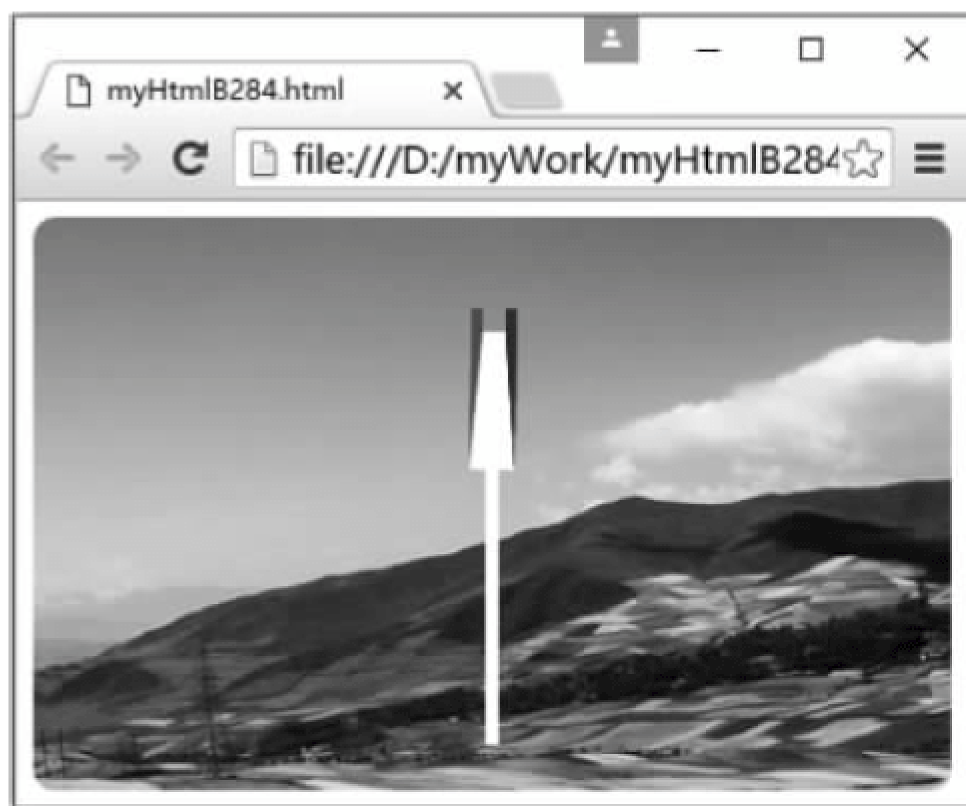


图 309-3

```
/＊绘制第一个桨叶＊/
.myBreeze { position: absolute; top: 30％; left: 50％;
           transform: translate(－50％, －50％) rotate(1deg);
           transform－origin: 50.5％ 62px; width:10px; height:10px;
           border－bottom: 60px solid ＃FFF; border－left: 5px solid green;
           border－right: 5px solid blue; }
```

此实例的源文件名是 myHtmlB284.html。

310　使用关键帧动画模拟彩虹和阴影的联动

此实例主要通过使用圆角产生的弧线制作彩虹并控制两个动画相同的持续时间，从而实现模拟彩虹和阴影的联动。当在 Google Chrome 浏览器中显示该页面时，彩虹左右摆动时下面的阴影也同时联动，如果彩虹达到左端，则下面与之对应的阴影最小，如图 310-1 所示；如果彩虹在中间，则下面与之对应的阴影最大。有关此实例的主要代码如下。

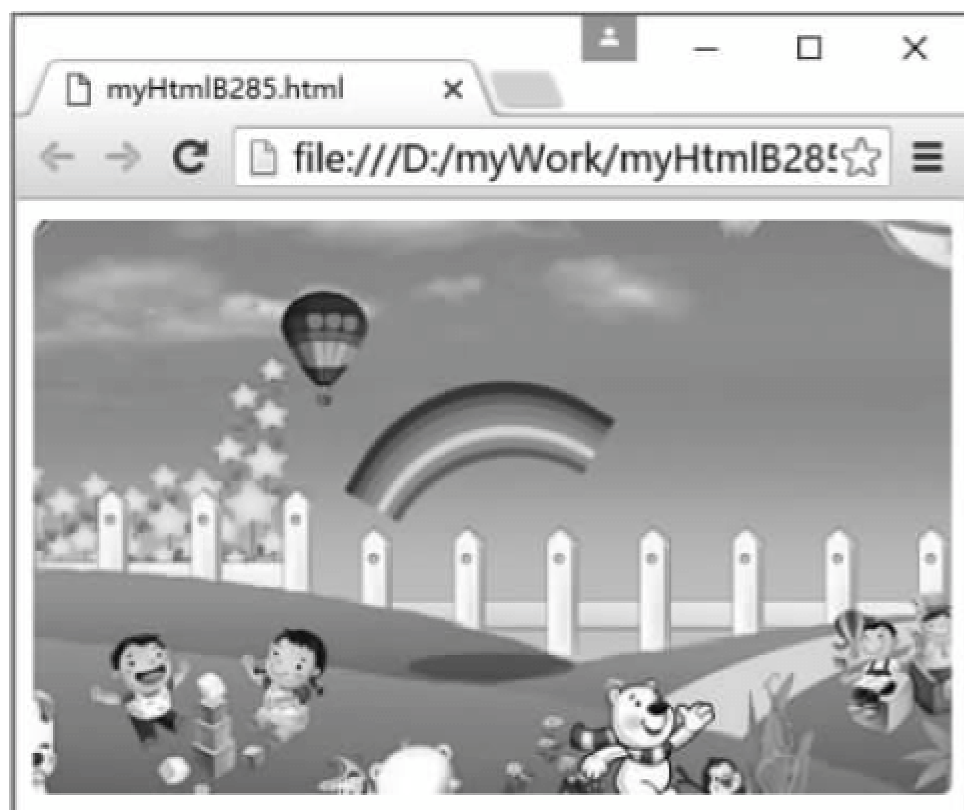


图 310-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox{ position: absolute; top: 50%; left: 50%;
          width: 400px; height: 250px; transform: translate(-50%, -50%);
          background:url(img/B281.jpg); border-radius: 5px;}
  .myRainbow{ position:absolute; top: 50%; left: 50%;
             transform: translate(-50%, -50%);}
  .myRainbow:before{ content:""; position:absolute; top: 50%;left: 50%;
        /*将矩形旋转45°,使左上角的圆角产生的弧线向上顶起*/
        transform: translate(-50%, -50%) rotate(45deg);
        /*设置矩形的宽度和高度,仅设置矩形左上角的圆角半径,其他3个角不显示*/
        height: 60px;width: 60px; border-radius: 60px 0 0 0;
        /*根据左上角的圆角产生的向上弧线产生多级彩色阴影,即彩虹*/
        box-shadow: #F44336 -2px -2px 0 1px, #FF9800 -4px -4px 0 3px,
                    #FFEB3B -6px -6px 0 5px, #8BC34A -8px -8px 0 7px,
                    #00BCD4 -10px -10px 0 9px, #2196F3 -12px -12px 0 11px,
                    #9C27B0 -14px -14px 0 13px;
        /*使用关键帧动画来回晃动彩虹*/
        animation: myRotate 5s ease-in-out infinite; }
  .myRainbow:after{ content: ""; position: absolute;
                   top: 70px;left: 50%;
                   height: 15px; width: 90px; background: rgba(0, 0, 0, 0.5);
                   border-radius: 50%;transform: translate(-50%, -50%);
                   /*使用关键帧动画缩放阴影*/
                   animation: myShadow 5s ease-in-out infinite; }
  @keyframes myRotate { 50% { transform: translate(-50%, -55%) rotate(30deg); }
           100% { transform: translate(-50%, -50%) rotate(45deg); } }
  @keyframes myShadow {50% {transform: translate(-50%, -50%) scale(0.8);
                           background: rgba(0, 0, 0, 0.2); }
           100% {transform: translate(-50%, -50%) scale(1);
                 background: rgba(0, 0, 0, 0.5); } }
 </style></head>
<body>
 <div class="myBox"><div class="myRainbow"></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation：myRotate 5s ease-in-out infinite 表示在5秒内按照 ease-in-out 模式完成关键帧动画 myRotate 指定的 rotate 改变元素(彩虹)的旋转角度(即左右晃动)，且永不停歇，此实例最奇妙的地方是使用圆角矩形的圆角弧线来制作圆弧形的彩虹；border-radius：60px 0 0 0 表示左上角的圆角半径是60px，其他3个角省略，因此可见的只有四分之一的圆弧形的彩虹。在CSS3中，border-radius 属性用于设置元素的圆角半径，该属性的语法格式如下。

```
border-radius: 1-4 length|%/1-4 length|%;
```

其中，length 表示以(像素)值设置圆角半径；%表示以百分比形式定义圆角半径；1-4的顺序分别是左上角、右上角、右下角、左下角，按此顺序设置每个角的半径值。如果省略 bottom-left，则与 top-right 相同；如果省略 bottom-right，则与 top-left 相同；如果省略 top-right，则与 top-left 相同。例如"border-radius：2em 1em 4em/0.5em 3em；"等价于以下代码。

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
```

```
border - bottom - right - radius: 4em 0.5em;
border - bottom - left - radius: 1em 3em;
```

此实例的源文件名是 myHtmlB285.html。

311 使用关键帧动画模拟夜空中闪烁的星星

此实例主要使用多级阴影产生多个星星，并在关键帧中设置不同的偏移位置，从而实现星星闪烁的动态效果。当在 Google Chrome 浏览器中显示该页面时，当月亮小幅度摆动时星星也出现一明一暗的闪烁效果，如图 311-1 所示。有关此实例的主要代码如下。

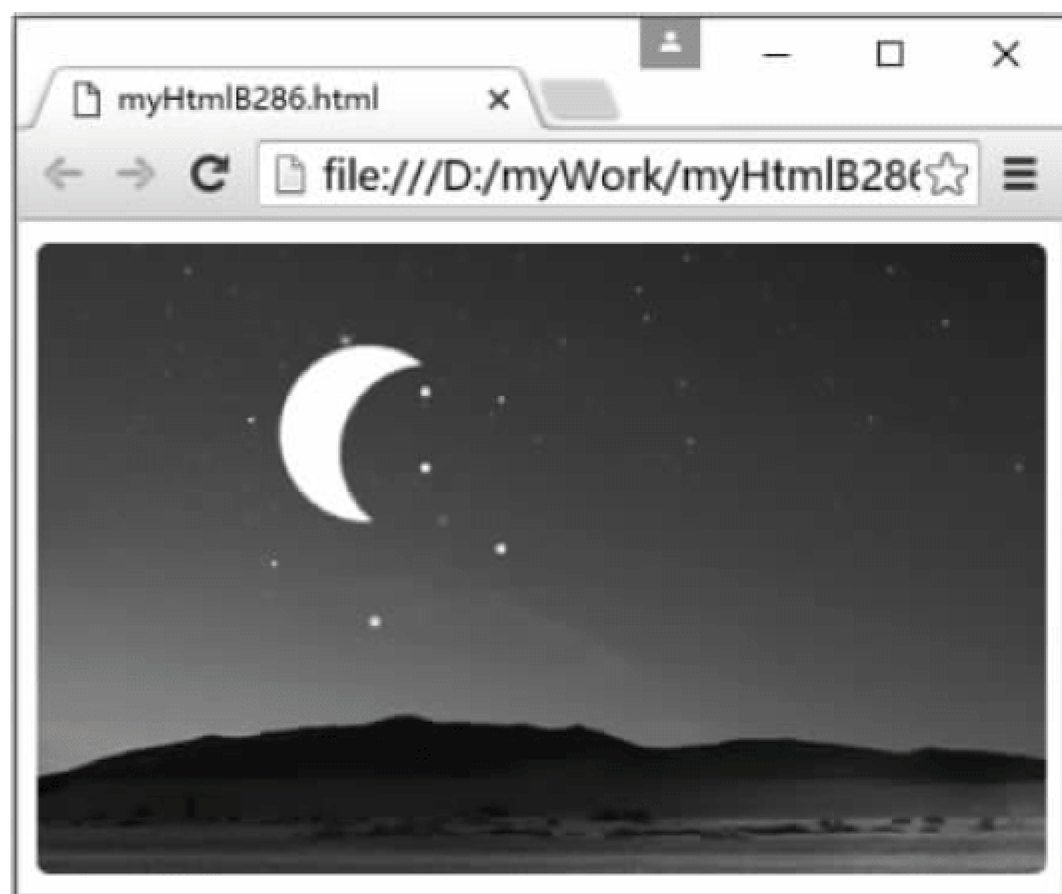


图 311-1

```
<! doctype html >< html >< head >< meta charset = "UTF - 8">
 < style type = "text/css">
   .myBox { position: absolute; top: 50 % ; left: 50 % ; width: 400px; height: 250px;
            transform: translate( - 50 % , - 50 % );background: url( img/B286.jpg);
            border - radius: 5px;}
  .myMoon { position: absolute;top: 20 % ;left: 30 % ;
           transform: translate( - 50 % , - 50 % ); height: 4px; width: 4px;border - radius: 50 % ;
           /* 通过阴影生成多个星星 */
           box - shadow: yellow - 26px 77px 0 - 1px, rgba(255, 255, 255, 0.1) - 36px 59px 0 - 1px, rgba
           (255, 255, 255, 0.1) - 28px 89px 0 - 1px, yellow - 35px 20px 0 - 1px, yellow 14px 100px, rgba
           (255, 255, 255, 0.1) 41px 60px, yellow 34px 39px, rgba(255, 255, 255, 0.1) 14px 45px 0 - 1px,
           yellow 64px 12px 0 - 1px, rgba(255, 255, 255, 0.1) 32px 96px 0 - 1px, yellow 64px 71px, rgba
           (255, 255, 255, 0.1) 60px 18px 0 - 1px,yellow 34px 9px, rgba(255, 255, 255, 0.1) - 26px 55px
           0 - 1px;
           /* 启动星星闪烁动画 */
           animation: myMoon_star 5s ease - in - out infinite; }
  .myMoon:before { content: ""; position: absolute; top: 20 % ; left: 50 % ;
      width: 70px; height: 70px; box - shadow: white - 25px 0;
      transform: rotate(15deg); border - radius: 50 % ;
      animation: myMoon 5s ease - in - out infinite; }
  /* 实现晃动的月亮 */
   @keyframes myMoon { 50 % { transform: rotate(45deg); } }
  /* 实现闪烁的星星 */
```

```
@keyframes myMoon_star { 50% { box-shadow: rgba(255, 255, 255, 0.1) -26px 77px 0 -1px, yellow -36px 59px 0 -1px, yellow -28px 89px 0 -1px, rgba(255, 255, 255, 0.1) -35px 20px 0 -1px, rgba(255, 255, 255, 0.1) 14px 100px, yellow 41px 60px, rgba(255, 255, 255, 0.1) 34px 39px, yellow 14px 45px 0 -1px, rgba(255, 255, 255, 0.1) 64px 12px 0 -1px, yellow 32px 96px 0 -1px, rgba(255, 255, 255, 0.1) 64px 71px, yellow 60px 18px 0 -1px, rgba(255, 255, 255, 0.1) 34px 9px, yellow -26px 55px 0 -1px; } }
</style></head>
<body><div class="myBox"><div class="myMoon"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation: myMoon_star 5s ease-in-out infinite 表示在 5 秒内按照 ease-in-out 模式完成关键帧 myMoon_star 中指定的一明一暗的星星闪烁动画，且永不停歇；animation: myMoon 5s ease-in-out infinite 表示在 5 秒内按照 ease-in-out 模式完成以关键帧 myMoon 中指定的角度旋转月亮的摆动动画，且永不停歇。此实例最奇妙的地方是在 box-shadow: white −25px 0 中通过弧线阴影产生的月亮，如果将此代码修改为 box-shadow: white 25px 0，则反向产生一个向右的月亮。

此实例的源文件名是 myHtmlB286.html。

312 使用关键帧动画模拟夜空中下落的流星雨

此实例主要在多个关键帧中使用 box-shadow 生成不同位置的阴影圆形（流星），然后使用动画不断改变每个流星的位置，从而实现夜空中流星雨下落的动态效果。当在 Google Chrome 浏览器中显示该页面时，流星（中间部分的黄点）从夜空中下落的效果如图 312-1 所示。有关此实例的主要代码如下。

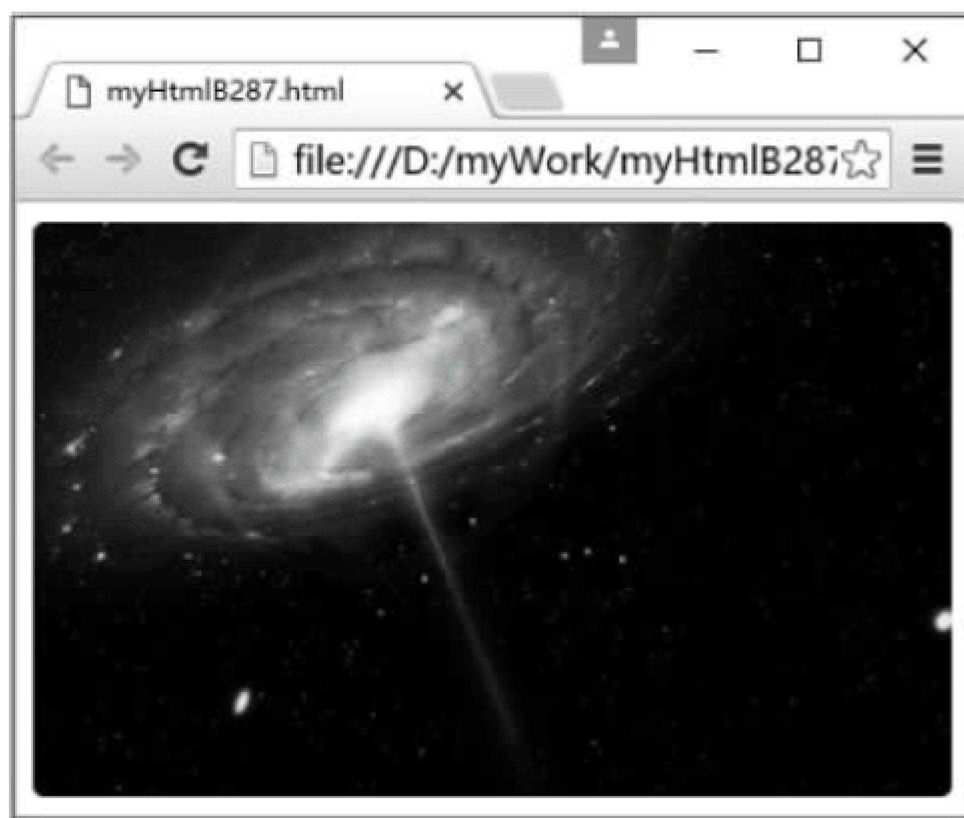


图 312-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { position: absolute; top: 50%; left: 50%;width: 400px; height: 250px;
          transform: translate(-50%, -50%);background: url(img/B287.jpg);
          border-radius: 5px;}
  .myMeteor { position: absolute; width: 3px; height: 3px; border-radius:50%;
      top: 30%; left: 50%; background: transparent; border-radius: 50%;
      /*启动关键帧中的流星雨动画*/
      animation: myMeteorRain 2s infinite linear; }
```

```
   @keyframes myMeteorRain { 0% {/* 通过创建多个阴影(圆)获得多个流星 */
      box-shadow:rgba(255, 255, 0, 0) -10px 30px, rgba(255, 255, 0, 0) 40px 40px,
                 rgba(255, 255, 0, 0.6) -50px 75px, rgba(255, 255, 0, 0.6) 55px 50px,
                 rgba(255, 255, 0, 0.6) -18px 100px, rgba(255, 255, 0, 0.6) 12px 95px,
                 rgba(255, 255, 0, 0.6) -31px 45px, rgba(255, 255, 0, 0.6) 30px 35px; }
 25% {box-shadow: rgba(255, 255, 0, 0.6) -10px 45px, rgba(255, 255, 0, 0.6) 40px 60px,
                  rgba(255, 255, 0, 0.6) -50px 90px, rgba(255, 255, 0, 0.6) 55px 65px,
                  rgba(255, 255, 0, 0) -18px 120px, rgba(255, 255, 0, 0) 12px 120px,
                  rgba(255, 255, 0, 0.6) -31px 70px, rgba(255, 255, 0, 0.6) 30px 60px; }
 26% {box-shadow: rgba(255, 255, 0, 0.6) -10px 45px, rgba(255, 255, 0, 0.6) 40px 60px,
                  rgba(255, 255, 0, 0.6) -50px 90px, rgba(255, 255, 0, 0.6) 55px 65px,
                  rgba(255, 255, 0, 0) -18px 40px, rgba(255, 255, 0, 0) 12px 20px,
                  rgba(255, 255, 0, 0.6) -31px 70px, rgba(255, 255, 0, 0.6) 30px 60px; }
 50% {box-shadow: rgba(255, 255, 0, 0.6) -10px 70px, rgba(255, 255, 0, 0.6) 40px 80px,
                  rgba(255, 255, 0, 0) -50px 100px, rgba(255, 255, 0, 0.6) 55px 80px,
                  rgba(255, 255, 0, 0.6) -18px 60px, rgba(255, 255, 0, 0.6) 12px 45px,
                  rgba(255, 255, 0, 0.6) -31px 95px, rgba(255, 255, 0, 0.6) 30px 85px; }
 51% { box-shadow:rgba(255, 255, 0, 0.6) -10px 70px,rgba(255, 255, 0, 0.6) 40px 80px,
                  rgba(255, 255, 0, 0) -50px 45px, rgba(255, 255, 0, 0.6) 55px 80px,
                  rgba(255, 255, 0, 0.6) -18px 60px, rgba(255, 255, 0, 0.6) 12px 45px,
                  rgba(255, 255, 0, 0.6) -31px 95px, rgba(255, 255, 0, 0.6) 30px 85px; }
 75% { box-shadow: rgba(255, 255, 0, 0.6) -10px 95px, rgba(255, 255, 0, 0.6) 40px 100px,
                   rgba(255, 255, 0, 0.6) -50px 60px, rgba(255, 255, 0, 0) 55px 95px,
                   rgba(255, 255, 0, 0.6) -18px 80px, rgba(255, 255, 0, 0.6) 12px 70px,
                   rgba(255, 255, 0, 0) -31px 120px, rgba(255, 255, 0, 0) 30px 110px;}
 76% { box-shadow: rgba(255, 255, 0, 0.6) -10px 95px, rgba(255, 255, 0, 0.6) 40px 100px,
                   rgba(255, 255, 0, 0.6) -50px 60px, rgba(255, 255, 0, 0) 55px 35px,
                   rgba(255, 255, 0, 0.6) -18px 80px, rgba(255, 255, 0, 0.6) 12px 70px,
                   rgba(255, 255, 0, 0) -31px 25px,rgba(255, 255, 0, 0) 30px 15px; }
 100% {box-shadow:rgba(255, 255, 0, 0) -10px 120px, rgba(255, 255, 0, 0) 40px 120px,
                  rgba(255, 255, 0, 0.6) -50px 75px, rgba(255, 255, 0, 0.6) 55px 50px,
                  rgba(255, 255, 0, 0.6) -18px 100px, rgba(255, 255, 0, 0.6) 12px 95px,
                  rgba(255, 255, 0, 0.6) -31px 45px, rgba(255, 255, 0, 0.6) 30px 35px; }}
  </style></head>
 <body><div class="myBox"><div class="myMeteor"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：rgba(255，255，0，0) −10px 30px，rgba(255，255，0，0) 40px 40px，rgba(255，255，0，0.6) −50px 75px，rgba(255，255，0，0.6) 55px 50px，rgba(255，255，0，0.6) −18px 100px，rgba(255，255，0，0.6) 12px 95px，rgba(255，255，0，0.6) −31px 45px，rgba(255，255，0，0.6) 30px 35px 用于一次性产生多个不同位置的流星(圆)，它主要是通过设置阴影的不同水平位置和垂直位置来实现的。在实例中有多个关键帧，在 animation 使用补间动画修补后它们在执行时就成了连续的动画，形成完整、流畅的动作。animation：myMeteorRain 2s infinite linear 表示在两秒内按照 linear 模式显示关键帧动画 myMeteorRain 中每个百分点指定的流星，且永不停歇。在动画开始后，除了通过不断改变位置产生流星下落的效果外，还有改变透明度让流星在下落时一会儿隐藏、一会儿显示的穿越动画效果。

此实例的源文件名是 myHtmlB287.html。

313 使用关键帧动画实现水波波纹扩散的效果

此实例主要在 animation 属性中使用 scale()方法改变大小，从而实现水波波纹一圈接一圈由里向外自动扩散的效果。当在 Google Chrome 浏览器中显示该页面时，圆形的图片将自动呈现水波波

纹一圈接一圈由里向外自动扩散的效果,如图 313-1 所示。有关此实例的主要代码如下。



图 313-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   .myWave{ position: absolute; top: 50%;left: 50%;transform: translate(-50%, -50%); width:
200px;height: 200px;background-image:url('img/B299.jpg');
background-position:center center; overflow:hidden; border-radius: 50%; box-shadow: 2px 2px 8px
black; }
   .myWave::before,.myWave::after{ content:"";position:absolute;
    background-image:url('img/B299.jpg');background-position:center center;
    border-radius:50%;}
   .myWave::before{ z-index:2; animation:myWave 5s ease-out infinite;}
   .myWave::after{ z-index:3; animation:myWave 5s ease-out 0.5s infinite;}
   @keyframes myWave{ 0%{ top: 50%;left: 50%;
    transform: translate(-50%, -50%) scale(1.5);width:30px;height:30px;}
            50%{ top: 50%;left: 50%; transform: translate(-50%, -50%) scale(1);
                 /*设置波纹扩散后的大小*/ width:240px;height:240px;}
            100%{ top: 50%;left: 50%; transform: translate(-50%, -50%) scale(1);
                   width:240px;height:240px; } }
 </style></head>
<body><div class="myWave"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,animation: myWave 5s ease-out infinite 表示在 5 秒内按照 ease-out 模式完成关键帧动画 myWave 指定的改变图像大小的动作,且永不停歇;animation: myWave 5s ease-out 0.5s infinite 实现的功能与前面的代码相同,但是有 0.5 秒的延迟,因此产生波纹一圈接一圈扩散的动态效果。

此实例的源文件名是 myHtmlB299.html。

314 使用关键帧动画模拟座椅在地板上滑动

此实例主要通过在 animation 属性中改变元素(座椅)与右边缘的距离模拟座椅在地板上的左右滑动效果。当在 Google Chrome 浏览器中显示该页面时,座椅将首先从左边滑向右边,如图 314-1 所示;然后再从右边滑向左边,如此往复,永不停歇。有关此实例的主要代码如下。

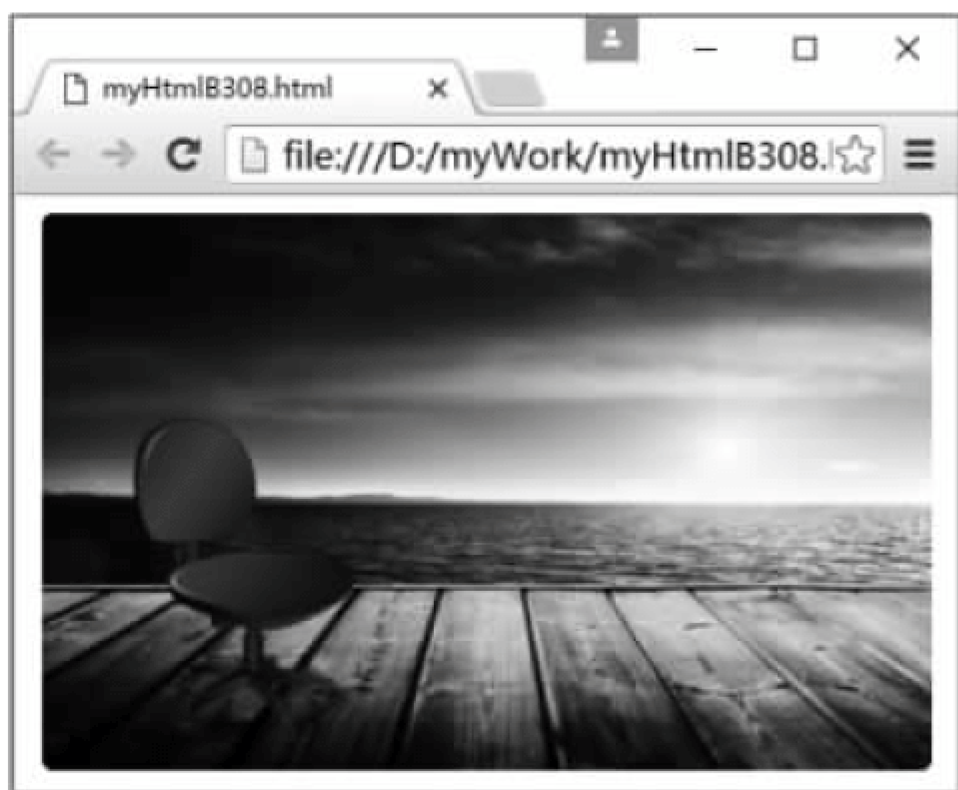


图 314-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { margin: 0 auto; width: 400px; height: 250px; border-radius: 5px;
       background-image: url("img/B308B.jpg"); background-size: 100% 100%; }
  img { position: relative; width: 150px; height: 150px; bottom: -90px;
        -webkit-animation: mySlide 10s linear infinite; }
  @-webkit-keyframes mySlide { 0% { right: 0;} 50% { right: -250px; }
                               100% { right: 0; } }
 </style></head>
<body><div><img src="img/B308A.png"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，right：－250px 表示座椅与右边缘的距离是－250px。在 CSS 中，right 属性规定元素的右边缘，该属性定义了定位元素右外边距边界与其包含块右边界之间的偏移，如果 position 属性值为 static，那么设置 right 属性不会产生任何效果。-webkit-animation：mySlide 10s linear infinite 表示在 10 秒内按照 linear 模式完成关键帧动画 mySlide 指定的改变座椅与右边缘的距离的动作，即座椅与右边缘的距离不断在 0 到－250，再返回 0 之间循环，永不停歇。

此实例的源文件名是 myHtmlB308.html。

315　使用关键帧动画模拟内、外圆环转动特效

此实例主要在 animation 属性中以动画的形式旋转元素并改变透明度，从而实现模拟内、外圆环错位连续转动的特效。当在 Google Chrome 浏览器中显示该页面时，内、外两个蓝色的圆环将错位旋转，且永不停歇，效果如图 315-1 所示。有关此实例的主要代码如下。

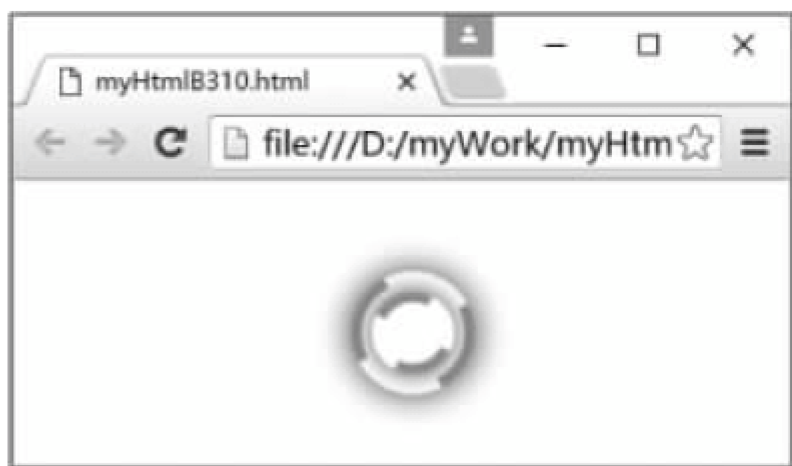


图 315-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*外层圆环设置*/
  #myOutCircle { background-color: rgba(0, 0, 0, 0); opacity: 0.9;
                 border: 5px solid rgba(0, 183, 229, 0.9);border-right: 5px solid rgba(0, 0, 0, 0);
                 border-left: 5px solid rgba(0, 0, 0, 0); border-radius: 50px;
                 box-shadow: 0 0 35px #2187E7;width: 50px; height: 50px; margin: 0 auto;
                 position: fixed;left: 50%; top: 50%; margin-left: -25px; margin-top: -25px;
                 -webkit-animation: OutCirclePulse 1s infinite ease-in-out;}
  /*外层圆环关键帧设置*/
  @-webkit-keyframes OutCirclePulse {
           0% {-webkit-transform: rotate(160deg);
               opacity: 0; box-shadow: 0 0 1px #505050; }
           50% { -webkit-transform: rotate(145deg); opacity: 1; }
           100% { -webkit-transform: rotate(-320deg); opacity: 0;} }
  /*内层圆环设置*/
  #myInCircle { background: rgba(0, 0, 0, 0) no-repeat center center;
                border: 5px solid rgba(0, 183, 229, 0.9); opacity: 0.9;
                /*产生上、下两个断开圆环*/
                border-top: 5px solid rgba(0, 0, 0, 0);
                border-bottom: 5px solid rgba(0, 0, 0, 0);
                border-radius: 50px; box-shadow: 0 0 15px #2187E7;
                width: 30px; height: 30px; margin: 0 auto; position: fixed;
                left: 50%; top: 50%;margin-left: -15px; margin-top: -15px;
                -webkit-animation: InCirclePulse 3s infinite linear; }
  /*内层圆环关键帧设置*/
  @-webkit-keyframes InCirclePulse { 0% { -webkit-transform: rotate(0deg);}
                                     100% { -webkit-transform: rotate(360deg); } }
 </style></head>
<body><div id="myOutCircle"></div><div id="myInCircle"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-right：5px solid rgba(0，0，0，0)用于设置外层圆环的右边框线(四分之一圆环)的线宽为5px，完全透明，即产生断开的效果；opacity：0表示完全透明，opacity：1表示完全不透明，外层圆环通过控制透明度决定自己在何时隐藏、何时显示，从而在旋转时产生闪烁的效果；rotate(160deg)表示沿着顺时针方向旋转160°；-webkit-animation：OutCirclePulse 1s infinite ease-in-out表示在1秒内按照ease-in-out模式完成关键帧动画OutCirclePulse指定的改变旋转角度和透明度的动作，且永不停歇。

此实例的源文件名是myHtmlB310.html。

316 使用关键帧动画实现带阴影渐变进度条

此实例主要在animation属性中以动画的形式改变元素的宽度，从而创建带阴影的渐变进度条。当在Google Chrome浏览器中显示该页面时，带渐变色的进度条将从左向右改变进度，且永不停歇，效果如图316-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { left: 10px; position: absolute; top: 20px; }
```

```
    .myProgress { background - color: #E5E9Eb;
                  height: 0.25em; position: relative; width: 20em; }
    /*绘制渐变进度条*/
    .myProgress - bar { background - image: linear - gradient(to right, #4CD964, #5AC8FA, #007AFF, #
                        34AADC, #5856D6, #FF2D55);
                        background - size: 20em 0.25em; height: 100%;
                        position: relative; animation: myWidth 3s linear infinite;}
    /*绘制渐变进度条下面的阴影*/
    .myProgress - shadow { background - image: linear - gradient(to bottom, #EAECEE, transparent);
               height: 4em; position: absolute; top: 100%;
               transform: skew(45deg); transform - origin: 0 0; width: 100%; }
    @ - webkit - keyframes myWidth {
       0%, 100% {transition - timing - function: cubic - bezier(1, 0, 0.65, 0.85);}
       0% { width: 0; } 100% { width: 100%; } }
   </style></head>
  <body><div class = "myBox">
   <div class = "myProgress"><div class = "myProgress - bar">
  <div class = "myProgress - shadow"></div></div></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，width：100%表示用百分比来定义宽度，百分比参照块宽度，在此实例中使用百分比最大的优势是盒子中的元素宽度始终随着盒子的宽度而改变；linear-gradient(to right, #4CD964, #5AC8FA, #007AFF, #34AADC, #5856D6, #FF2D55)用于以从左向右的方向创建线性渐变的背景；animation：myWidth 3s linear infinite 表示在 3 秒内按照 linear 模式完成关键帧动画 myWidth 指定的改变宽度的动作，且永不停歇。

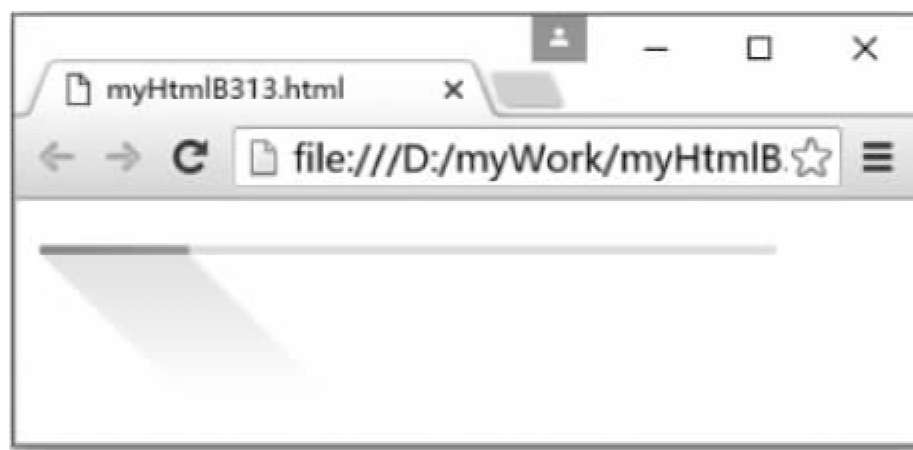


图 316-1

此实例的源文件名是 myHtmlB313.html。

317 使用关键帧动画实现当前按钮标题的闪烁显示

此实例主要使用关键帧动画设置按钮标题的 color 属性，从而实现当鼠标指针悬浮在按钮上时按钮标题不断闪烁的效果。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第一个按钮上，则该按钮的标题“奇妙数学的 100 个重大突破”将以黑色、青色、绿色的顺序交替闪烁，如图 317-1 所示；如果鼠标指针悬浮在其他按钮上，则该按钮的标题也将以黑色、青色、绿色的顺序交替闪烁。有关此实例的主要代码如下。



图 317-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a, a:visited { outline: none; color: black; }
  .myButton { display: inline-block; min-width: 250px;
             width: 60%; height: 20px; font-size: 18px;
             color: black;text-decoration: none !important;
             padding: 9px 30px;
             box-shadow: 0 5px 5px darkgrey;
             border-radius: 10px; background-color: grey; }
  .myButton:hover { -webkit-animation: myBlink 1s infinite ease; }
  /* 设置关键帧动画的闪烁颜色 */
  @-webkit-keyframes myBlink { 0% { color: black; }
                          40% { color: cyan; } 70% { color: lightgreen;} }
 </style></head>
<body>
<p align="center"><a href="#" class="myButton">奇妙数学的 100 个重大突破</a></p>
<p align="center"><a href="#" class="myButton">科学美国人趣味数学集锦</a></p>
<p align="center"><a href="#" class="myButton">改变世界的 134 个概率统计故事</a></p></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-decoration：none ！ important 用于取消超链接的下画线。! important 是 CSS1 定义的语法，作用是提高指定样式规则的应用优先权。其语法格式为{ cssRule ！ important }，即它写在 CSS 定义的最后面，例如 box{color：red ！ important;}。在默认情况下，CSS 规则按级层覆盖，例如在.css 文件中的定义可以被 style 标签的定义覆盖，反之则不行。然而，对于覆盖平衡而言，加上一个"！ important"就优先于正常的 CSS 规则。40% { color：cyan; }表示在动画期间的 40%时点上按钮标题的颜色是 cyan。-webkit-animation：myBlink 1s infinite ease 表示当前按钮的标题将在 1 秒内以 ease 模式执行 myBlink 关键帧中的改变颜色的动作，且永不停歇。

此实例的源文件名是 myHtmlB325.html。

318　使用关键帧动画实现圆环转圈加载的特效

此实例主要通过关键帧动画控制元素的 transform 属性指定的 rotate()方法实现圆环转圈加载的特效。当在 Google Chrome 浏览器中显示该页面时，圆环在转圈的过程中将按照顺序改变颜色，且永不停歇，如图 318-1 所示。有关此实例的主要代码如下。



图　318-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox{}
  /*设置内圆样式*/
  .myBox .inner_circle { position: relative; width: 120px; height: 120px;
                       background-color: white; border-radius: 50%;margin-top: -135px;
                       line-height: 120px;z-index: 999;}
   /*设置外圆样式*/
   .myBox .outer_circle { position: relative;width: 150px; height: 150px;
                        border-radius: 50%; background-color: cyan;}
   /*设置外圆与内圆之间的转圈圆(环)的样式*/
   .myBox .rotate_circle { position: relative; border-radius: 150px 0 0 150px;
                         width: 75px; height: 150px;background-color: red; margin-top: -150px;
                         margin-left: -75px; z-index: 222; -webkit-transform-origin: 75px 75px;
                         -webkit-animation: myRotate 5s infinite linear; }
   @-webkit-keyframes myRotate{ 0% { -webkit-transform: rotate(0deg);
                                      background-color: red; }
   50% { -webkit-transform: rotate(180deg); background-color: green; }
   100% { -webkit-transform: rotate(360deg); background-color: blue; } }
 </style></head>
<body><div class="myBox" align="center">
    <div class="outer_circle"></div><div class="rotate_circle"></div>
    <div class="inner_circle">正在加载...</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，z-index属性用于设置元素的堆叠顺序，拥有更高堆叠顺序的元素总是处于堆叠顺序较低的元素的前面，由于此实例是由3个不同半径的圆饼按照z-index属性值的大小堆叠在一起，转圈效果只需要中间的圆饼(环)转动即可；-webkit-transform：rotate(180deg)表示顺时针旋转180°；-webkit- transform-origin：75px 75px用于设置旋转元素(圆饼)的基点位置(75px 75px)；-webkit-animation：myRotate 5s infinite linear表示在5秒内以线性方式完成关键帧动画myRotate的旋转动作，且永不结束。

此实例的源文件名是myHtmlB326.html。

319　使用关键帧动画实现环绕矩形转动特效

此实例主要通过关键帧动画控制clip属性指定的裁剪矩形实现环绕矩形的转动特效。当在Google Chrome浏览器中显示该页面时，一条黄色的线条将按照逆时针方向环绕矩形图像做逆时针转动，如图319-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  html, body { margin: 0; padding: 0; background: #101010;}
  .myBox { width: 300px; margin: 50px auto; }
  /*设置图像盒子的大小*/
  .myImage { position: relative; width: 200px; height: 200px; }
  /*在图像盒子的外面嵌套黄色边框线的矩形*/
```

```
.myImage:after { content: " "; display: block; position: absolute; z-index: 10;
                width: 220px; height: 220px; top: -10px; left: -10px; border: 2px solid yellow;
                box-sizing: border-box; -webkit-animation: myClip 2s infinite linear; }
/* 使用关键帧裁剪黄色边框线的矩形 */
@keyframes myClip { 0%, 100% { clip: rect(0px, 220px, 220px, 217px); }
 25% { clip: rect(0px, 220px, 3px, 0px); } 50% { clip: rect(0px, 3px, 220px, 0px); }
 75% { clip: rect(217px, 220px, 220px, 0px); } }
img { width: 200px; height: 200px; border-radius: 5px; }
</style></head>
<body><div class="myBox" align="center">
   <div class="myImage"><img src="img/B328.jpg"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-sizing：border-box 用于控制边框盒。在 CSS3 中，box-sizing 属性允许以特定的方式定义匹配某个区域的特定元素。例如需要并排放置两个带边框的框，可将 box-sizing 设置为 border-box，这样可使浏览器呈现出带有指定宽度和高度的框，并把边框和内边距放入框中。box-sizing 属性主要支持下列两个属性值。

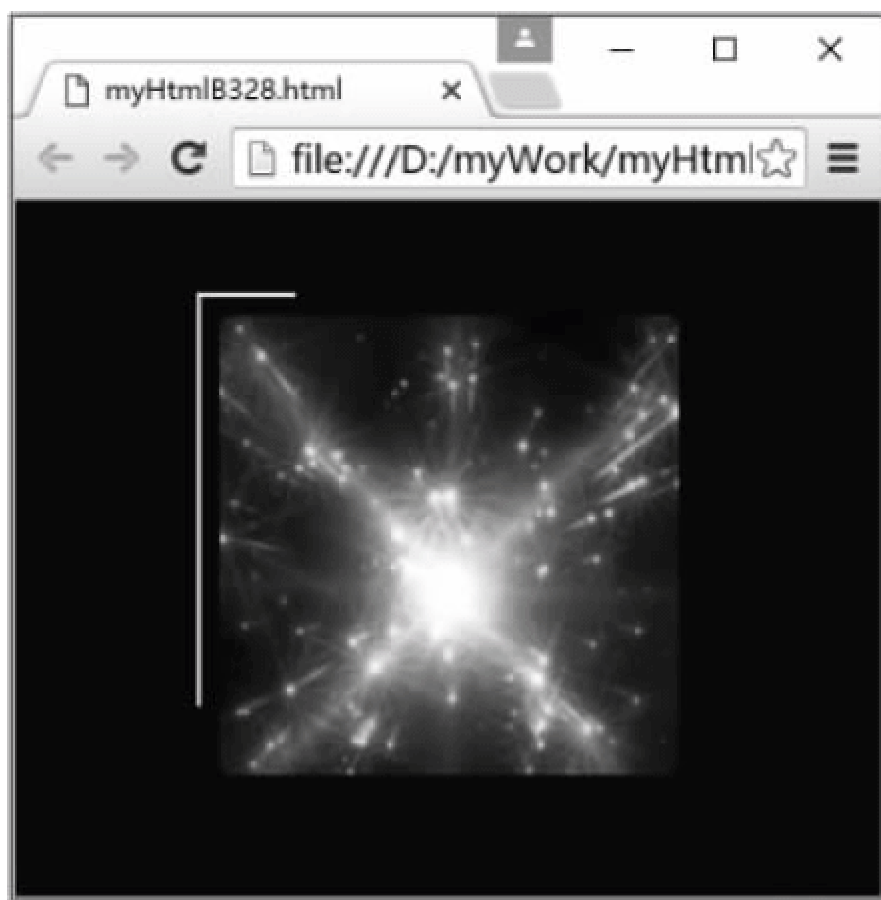


图　319-1

（1）content-box：该属性值表示宽度和高度分别应用到元素的内容框，在宽度和高度之外绘制元素的内边距和边框。

（2）border-box：该属性值表示为元素设定的宽度和高度决定了元素的边框盒。也就是说，为元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。

clip 属性用于剪裁绝对定位元素。也就是说，当一幅图像的尺寸大于包含它的元素时会发生什么呢？clip 属性允许规定一个元素的可见尺寸，这样此元素就会被修剪并显示为这个形状。对于一个绝对定义元素，在这个矩形内的内容才可见。出了这个剪裁区域的内容会根据 overflow 的值来处理。剪裁区域可能比元素的内容区大，也可能比内容区小。在此实例中，clip：rect(0px，220px，3px，0px)中的各参数值对应的方位是 rect (top，right，bottom，left)。

-webkit-animation：myClip 2s infinite linear 表示在两秒内以线性方式完成关键帧动画 myClip 的裁剪动作，且永不结束。

此实例的源文件名是 myHtmlB328.html。

320　使用关键帧动画实现凸轮滑动闪烁特效

此实例主要通过关键帧动画控制 text-shadow 属性产生的阴影特效实现文字闪烁时产生类似于凸轮滑动轨迹（如蒸汽火车的火车轮子连杆）的动态效果。当在 Google Chrome 浏览器中显示该页面时，“炫酷实例集锦”将根据关键帧所设置的阴影特效轮番闪烁，如图 320-1 所示。有关此实例的主要代码如下。



图 320-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  p { text-shadow: 2px 2px cyan;font-size:50px;
      -webkit-animation: myBlink 0.5s infinite; }
  @-webkit-keyframes myBlink { 0%, 100% { text-shadow: 2px 2px cyan, 3px 3px cyan; }
    25% { text-shadow: 2px -2px lightgreen, 3px -3px lightgreen; }
    50% { text-shadow: -2px -2px blue, -3px -3px blue; }
    75% { text-shadow: -2px 2px yellow, -3px 3px yellow; } }
 </style></head>
<body><div align="center"><p>炫酷实例集锦</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,“text-shadow: -2px -2px blue, -3px -3px blue”对应的是“text-shadow:水平阴影的位置 垂直阴影的位置 阴影的颜色,水平阴影的位置 垂直阴影的位置 阴影的颜色”;-webkit-animation: myBlink 0.5s infinite 表示在 500 毫秒内以默认方式完成关键帧动画 myBlink 中预置的设置阴影颜色和位置的变化动作,且永不结束。

此实例的源文件名是 myHtmlB329.html。

321 使用关键帧动画模拟立方体的 3D 旋转特效

此实例主要通过设置-webkit-perspective、-webkit-transform-origin、-webkit-transform-style 等属性模拟立方体的 3D 旋转特效。当在 Google Chrome 浏览器中显示该页面时,立方体将在当前空间中不断变换角度和颜色显示,且永不停歇,效果如图 321-1 所示。有关此实例的主要代码如下。

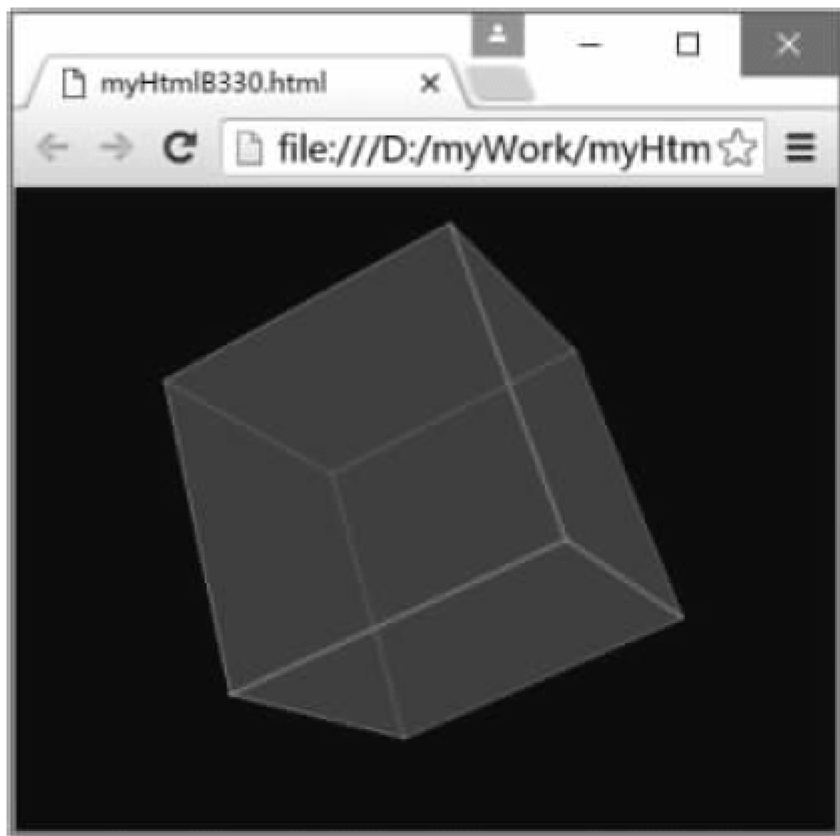


图 321-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body { background: #1A1A1A; -webkit-perspective: 600px; }
  .myBox { width: 150px; height: 150px;position: relative; top: 60px; left: 50%;
          margin-left: -75px; -webkit-transform-style: preserve-3d;
          -webkit-transform: rotateY(0deg) translateZ(-200px);
          -webkit-animation: myRotate 5s linear infinite; }
  .myFace { width: 150px; height: 150px;position: absolute; opacity: 0.45;
          background: #FF033E; -webkit-transform-origin: 50% 50%;
          border: 1px solid white; }
  .myFace-front { -webkit-transform: translateZ(75px);
                 -webkit-animation: changeColor 5s linear infinite;}
  .myFace-left {-webkit-transform: rotateY(90deg) translateZ(75px);
               -webkit-animation: changeColor 5s linear infinite;}
  .myFace-back { -webkit-transform: translateZ(-75px);
                -webkit-animation: changeColor 5s linear infinite;}
  .myFace-right {-webkit-transform: rotateY(90deg) translateZ(-75px);
                -webkit-animation: changeColor 5s linear infinite;}
  .myFace-top {-webkit-transform: rotateX(90deg) translateZ(-75px);
              -webkit-animation: changeColor 5s linear infinite;}
  .myFace-bottom { -webkit-transform: rotateX(90deg) translateZ(75px);
                  -webkit-animation: changeColor 5s linear infinite;}
  /*设置立方体的旋转关键帧*/
  @-webkit-keyframes myRotate {
    0% { -webkit-transform: rotateX(0deg) rotateY(0deg) rotateZ(0deg);}
    100% {-webkit-transform: rotateX(360deg) rotateY(360deg) rotateZ(360deg)}}
  /*设置立方体每面的颜色变化关键帧*/
  @-webkit-keyframes changeColor { 0%, 100% { background: cyan; }
            25% { background: darkgreen; } 50% { background: red; } }
 </style></head>
<body><div class="myBox">
 <div class="myFace myFace-front"></div><div class="myFace myFace-left"></div>
 <div class="myFace myFace-back"></div><div class="myFace myFace-right"></div>
 <div class="myFace myFace-top"></div><div class="myFace myFace-bottom"></div></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-perspective：600px 定义立方体和视图的距离是 600px，perspective 属性通常与 perspective-origin 属性同时使用；-webkit-transform-origin：50% 50%用于定义立方体的每个面在 X 轴和 Y 轴的位置都是 50%；-webkit-transform-style：preserve-3d 表示子元素(立方体的每个面)将保留其 3D 位置；-webkit-transform：rotateY(90deg) translateZ(75px)表示立方体的左面沿 Y 轴旋转 90°，沿 Z 轴平移 75px；-webkit-animation：myRotate 5s linear infinite 表示在5 秒内以线性方式完成 myRotate 所指定的旋转动作，且永不停歇。

此实例的源文件名是 myHtmlB330.html。

322 使用关键帧动画模拟多个立方体的旋转

此实例主要通过设置 perspective、transform-style 等属性模拟多个立方体沿着螺旋路径旋转的特效。当在 Google Chrome 浏览器中显示该页面时，12 个立方体(金条)将环绕螺旋线旋转，且永不停歇，如图 322-1 所示。有关此实例的主要代码如下。

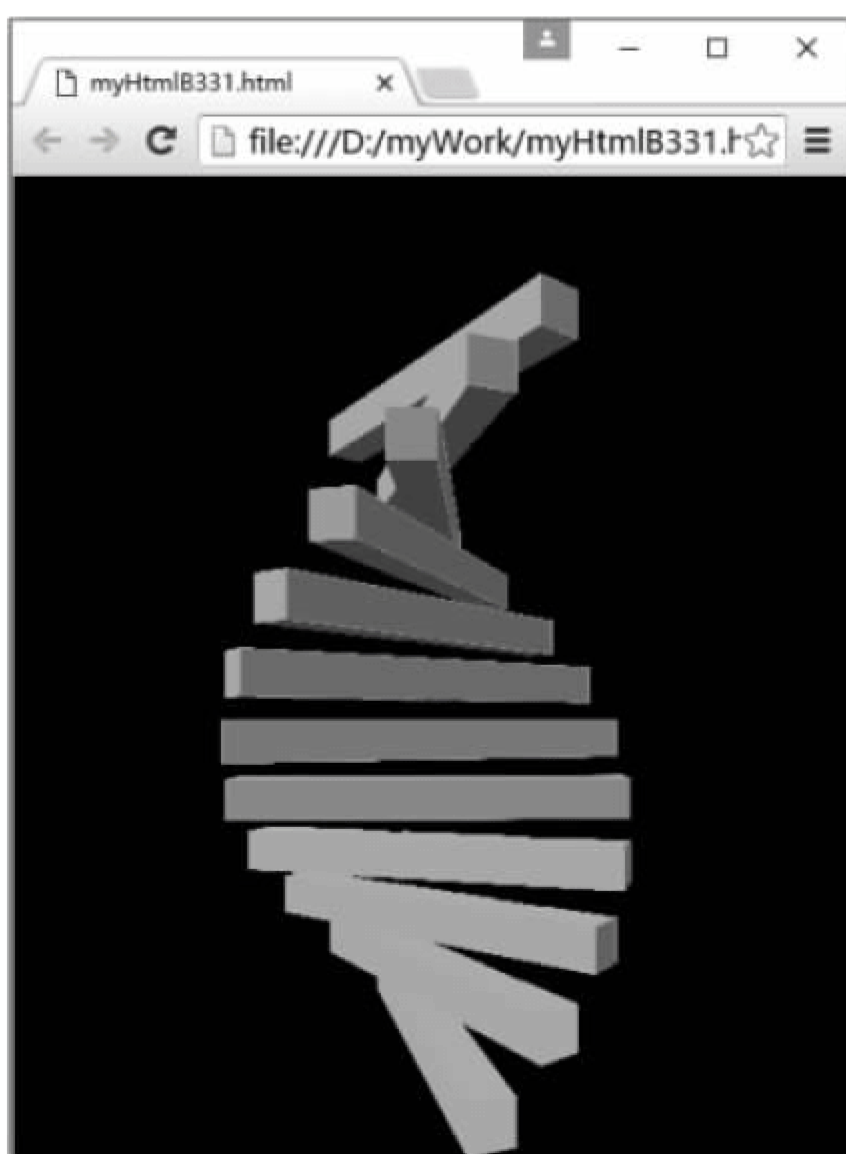


图 322-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   for (var a=1; a<=12; a++) {//添加 12 根金条
    $(".myBox").append("<div class='myCuboid'></div>");
    for (var b=1; b<=6; b++) {
      $(".myBox .myCuboid").append("<div class='front'></div><div class='left'> </div><div class=
'back'></div><div class='right'></div><div class='top'></div> <div class='bottom'></div>");
  } }
  for (var i=1; i<=12; i++) {//设置每根金条的动画参数
    $(".myCuboid:nth-child("+i+")").css("-webkit-animation-delay", -(1.9-0.1*(i-1))+"s");
    for (var j=1; j<=4; j++) {
     $(".myCuboid:nth-child("+i+") div:nth-child("+j+")").css("-webkit- animation-delay",
-(3.4-0.1*(i-1))+0.5*(j-1)+"s");
} } });
 </script>
 <style type="text/css">
  body{ background-color: black; }
  /*设置盒子的基本样式*/
  .myBox { position: relative; top: 90px;left:50px; margin: 0 auto;
          width: 300px; perspective: 400px; transform-style: preserve-3d; }
  /*设置金条的基本样式*/
  .myCuboid { position: relative; height: 20px; width: 200px;
             padding-bottom: 10px; transform-style: preserve-3d;
             -webkit-animation: myRotate 2s infinite linear; }
  .myCuboid div { position: absolute;background-color: #8A5B0F;
                 backface-visibility: hidden; }
  /*设置金条前面、左面、后面、右面的颜色动画*/
  .myCuboid .front, .myCuboid .left, .myCuboid .back, .myCuboid .right {
                         animation: myColor 1s infinite linear alternate; }
```

```
    /*设置金条前面、后面的尺寸*/
    .myCuboid .front, .myCuboid .back { height: 20px; width: 200px; }
    /*设置金条下面、上面的尺寸*/
    .myCuboid .bottom, .myCuboid .top { height: 20px; width: 200px; }
    /*设置金条左面、右面的尺寸*/
    .myCuboid .left, .myCuboid .right { height: 20px; width: 20px; }
    /*设置金条前面的平移尺寸*/
    .myCuboid .front { transform: translateZ(10px); }
    /*设置金条后面的旋转角度和平移尺寸*/
    .myCuboid .back { transform: rotateY(180deg) translateZ(10px); }
    /*设置金条上面的旋转角度和平移尺寸*/
    .myCuboid .top { transform: rotateX(90deg) translateZ(10px);
                    background-color: #F0C175; }
    /*设置金条下面的旋转角度和平移尺寸*/
    .myCuboid .bottom { transform: rotateX(-90deg) translateZ(10px); }
    /*设置金条左面的旋转角度和平移尺寸*/
    .myCuboid .left { transform: rotateY(-90deg) translateZ(10px); }
    /*设置金条右面的旋转角度和平移尺寸*/
    .myCuboid .right { transform: rotateY(90deg) translateZ(190px); }
    /*设置金条旋转动画的关键帧*/
    @keyframes myRotate { from { transform: rotateY(0deg); }
                          to { transform: rotateY(360deg); } }
  /*设置金条颜色动画的关键帧*/
    @keyframes myColor { from { background-color: red; }
                         to { background-color: yellow; } }
  </style></head>
 <body><div class='myBox'></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，perspective：400px 用于定义立方体（金条）与视图的距离为 400px；transform-style：preserve-3d 表示子元素将保留其 3D 位置；transform：rotateY(－90deg) translateZ(10px)表示立方体沿 Y 轴逆时针旋转 90°，沿 Z 轴平移 10px；-webkit-animation：myRotate 2s infinite linear 表示在两秒内以线性方式完成 myRotate 所指定的旋转动作，且永不停歇；animation：myColor 1s infinite linear alternate 表示在 1 秒内以线性方式完成 myColor 所指定的颜色变化动作，且轮流交替进行，永不停歇。

此实例的源文件名是 myHtmlB331.html。

323 使用关键帧动画模拟小圆点加载的状态

此实例主要在关键帧动画中使用 scale()方法缩放小圆点，从而模拟 Windows 10 操作系统启动时大小变化的小圆点加载进度状态。当在 Google Chrome 浏览器中显示该页面时，大小变化的小圆点将以顺时针方向转圈形式指示加载进度状态，效果如图 323-1 所示。有关此实例的主要代码如下。

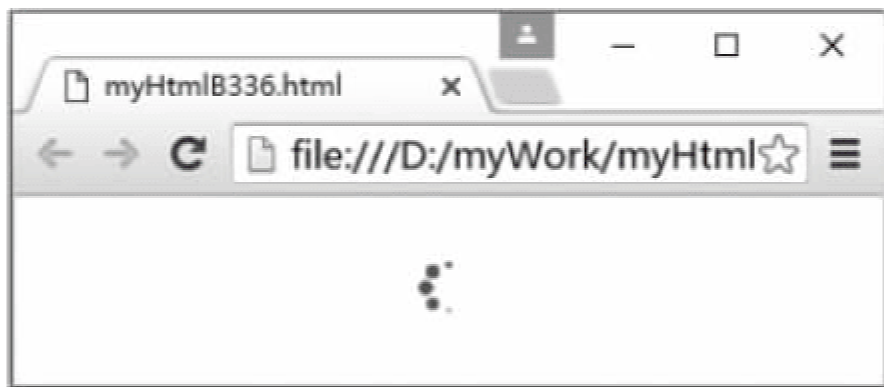


图 323-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   for(var i=1;i<=4;i++){
    for(var j=1;j<=4;j++){ $(".myBox"+i+" .circle"+j+"").css("animation-delay", ((-1.2+
                                  0.1*(i-1))+0.3*(j-1))+"s");
  } } });
 </script>
 <style type="text/css">
  /*设置转圈的样式*/
  .myContainer { margin: 30px auto; width: 20px; height: 20px; position: relative; }
  /*设置小圆点的样式*/
  .myBox1>div, .myBox2>div, .myBox3>div { width: 6px; height: 6px;
           background-color: red; border-radius: 100%; position: absolute;
           animation: myScale 1.2s infinite ease-in-out; }
  .myContainer .myContainer-myBox { position: absolute;
                                       width: 100%; height: 100%; }
  .myBox2 { transform: rotateZ(45deg); }
  .myBox3 { transform: rotateZ(90deg); }
  .circle1 { top: 0;left: 0; } /*在左上角显示*/
  .circle2 { top: 0; right: 0; } /*在右上角显示*/
  .circle3 { right: 0; bottom: 0;} /*在右下角显示*/
  .circle4 { left: 0; bottom: 0;} /*在左下角显示*/
  @keyframes myScale { 0%, 80%, 100% { transform: scale(0.0) } /*小圆点消失*/
                        40% { transform: scale(1.0) } /*小圆点出现*/}
 </style></head>
<body><div class="myContainer">
  <div class="myContainer-myBox myBox1">
    <div class="circle1"></div><div class="circle2"></div>
    <div class="circle3"></div><div class="circle4"></div></div>
  <div class="myContainer-myBox myBox2">
    <div class="circle1"></div><div class="circle2"></div>
    <div class="circle3"></div><div class="circle4"></div></div>
 <div class="myContainer-myBox myBox3">
    <div class="circle1"></div><div class="circle2"></div>
    <div class="circle3"></div><div class="circle4"></div></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform: scale(0.0)表示将小圆点(6,6)缩小至0，即消失，transform: scale(1.0)表示将小圆点(6,6)缩放至原始大小，即显示；animation-delay属性定义动画执行前等待的时间，以秒或毫秒计，且允许为负值，−2s表示动画马上开始，但跳过两秒进入动画；animation: myScale 1.2s infinite ease-in-out表示在1200毫秒内以ease-in-out方式完成关键帧动画myScale的缩放动作，且永不结束。

此实例的源文件名是myHtmlB336.html。

324 使用关键帧动画实现以多色圆环模拟加载进度

此实例主要在关键帧动画中重置模糊圆的位置，从而实现以转圈的多色圆环模拟加载进度状态的效果。当在Google Chrome浏览器中显示该页面时，红、蓝、黄、绿等色将在圆环上以顺时针方向转圈形式指示加载进度状态，效果如图324-1所示。有关此实例的主要代码如下。



图 324-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body { background-color: gray;}
  /*设置外圆样式*/
  body>div { position: relative; background-color: darkgrey; width: 200px;
             height: 200px; border-radius: 100%; overflow: hidden; margin: 20px auto; }
  /*设置内圆样式*/
  /*父元素(body)的第2个子元素(div)*/
  body>div:nth-child(2) { width: 160px; height: 160px; top: -200px;
                          line-height: 160px; text-align: center;background-color: pink; }
  /*设置模糊圆的样式*/
  body>div div { width: 155px; height: 155px;border-radius: 100%;
                 -webkit-filter: blur(20px);position: absolute; }
  /*父元素(div)的第1个子元素(div)*/
  body>div div:nth-child(1) { background-color: red;
                               -webkit-animation: myMove 4s infinite linear; -webkit-animation-
                              delay: 4.8s; }
  /*父元素(div)的第2个子元素(div)*/
  body>div div:nth-child(2) {background-color: green;
                              -webkit-animation: myMove 3s infinite linear; }
  /*父元素(div)的第3个子元素(div)*/
  body>div div:nth-child(3) { background-color: blue;
                               -webkit-animation: myMove 6s infinite linear; -webkit-animation-
                              delay: 3.2s;}
  /*父元素(div)的第4个子元素(div)*/
  body>div div:nth-child(4) { background-color: yellow;
                               -webkit-animation: myMove 5s infinite linear; -webkit-animation-
                              delay: 1.6s; }
  @-webkit-keyframes myMove {
            0%, 100% { top: 0%; left: 0%; } 25% { top: 0%; left: 50%; }
            50% { top: 50%;left: 50%; } 75% { top: 50%;left: 0%; } }
 </style></head>
<body><div><div></div><div></div><div></div><div></div></div>
<div>加载中,请稍候...</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，nth-child(n)选择器匹配属于其父元素的第 n 个子元素，而不论元素的类型，n 可以是数字、关键字或公式，例如 body > div:nth-child(2)即

表示 body 中的第 2 个 div 元素；-webkit-animation：myMove 4s infinite linear 表示在 4 秒内以 linear 方式完成关键帧动画 myMove 的转圈动作，且永不结束。

此实例的源文件名是 myHtmlB337.html。

325 使用关键帧动画模拟足球射门的动作变化

此实例主要通过在关键帧动画中同时执行旋转、平移、缩放动作模拟足球射门的变化动作。当在 Google Chrome 浏览器中显示该页面时，足球在射门前的状态如图 325-1 的左下角所示，在踢球后，足球向右上角的球门飞去时将发生旋转、移动以及远去的视觉变化。有关此实例的主要代码如下。

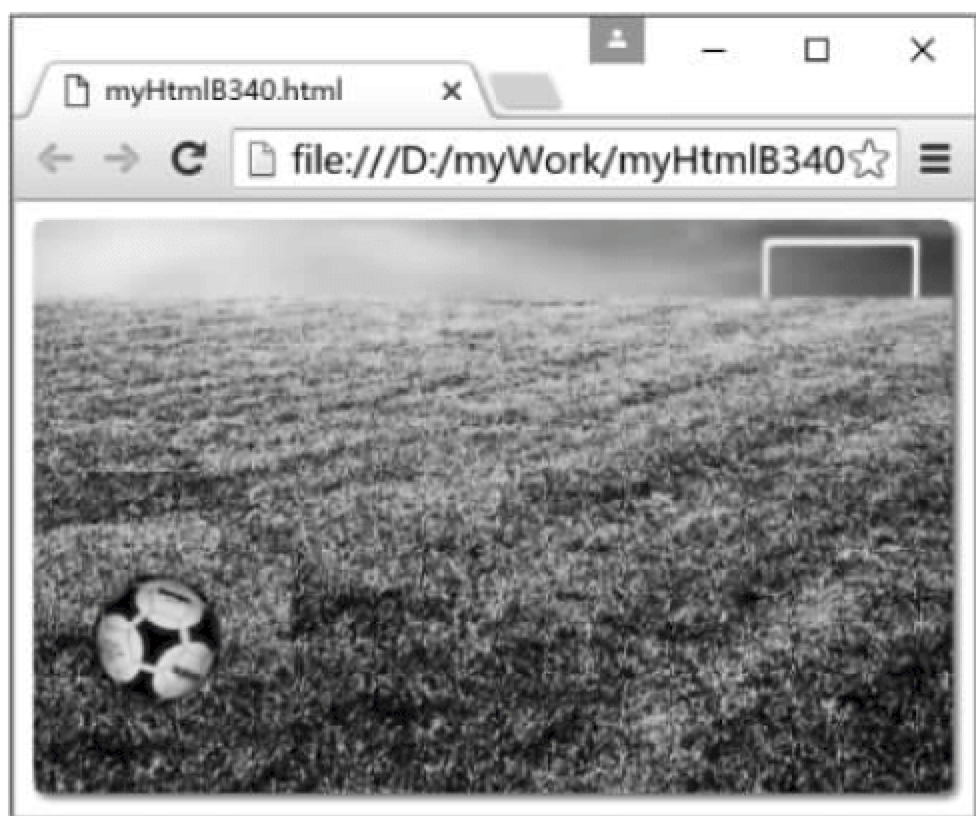


图 325-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   /*设置草坪的样式*/
   .myBox { width: 400px; height: 250px;background-image: url("img/B340.jpg");
            background-size: 100% 100%;border-radius: 5px;box-shadow: 2px 2px 5px gray; }
   /*设置足球的基本样式*/
   .myBall { position: absolute; left: 15px; top: 170px; width: 60px; height: 60px;
             border-radius: 50%;background-repeat: no-repeat;
             background-image: url(img/B255.png);
             -webkit-animation: myRoll 5s ease-in-out infinite;
             background-size: 100% 100%; }
   /*设置足球射门时的翻滚旋转动作*/
   @-webkit-keyframes myRoll {
          0%, 100% { -webkit-transform: translateX(0px) translateY(0px) rotateZ(0deg) scale(1); }
          99% { -webkit-transform: translateX(320px) translateY(-165px) rotateZ(360deg) scale(0.3); } }
 </style></head>
<body><div class="myBox"><div class="myBall"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，translateX(0px)和 translateX(320px)表示足球(div)将从 0px 位置沿着 X 轴平移到 320px 位置；translateY(0px)和 translateY(-165px)表示足球(div)将从 0px 位置沿着 Y 轴平移到-165px 位置，即从左下角沿对角线移动到右上角；rotateZ(0deg)和 rotateZ(360deg)表示足球将围绕 Z 轴从 0°旋转至 360°，即一圈；scale(0.3)表示在足球到达右上角的过程中将其大小逐渐缩小到原来的 30%；myRoll 5s ease-in-out infinite 表

示在 5 秒内完成关键帧动画 myRoll 中的旋转、平移、缩放动作，且永不停歇。

此实例的源文件名是 myHtmlB340.html。

326　使用关键帧动画模拟雷达的动态扫描效果

此实例主要通过在关键帧动画中执行旋转动作模拟雷达扫描的动态效果。当在 Google Chrome 浏览器中显示该页面时，绿色渐变的扫描扇形将围绕中心做顺时针的旋转动作，如果发现目标，则会出现一个亮点闪烁提示，效果如图 326-1 所示。有关此实例的主要代码如下。

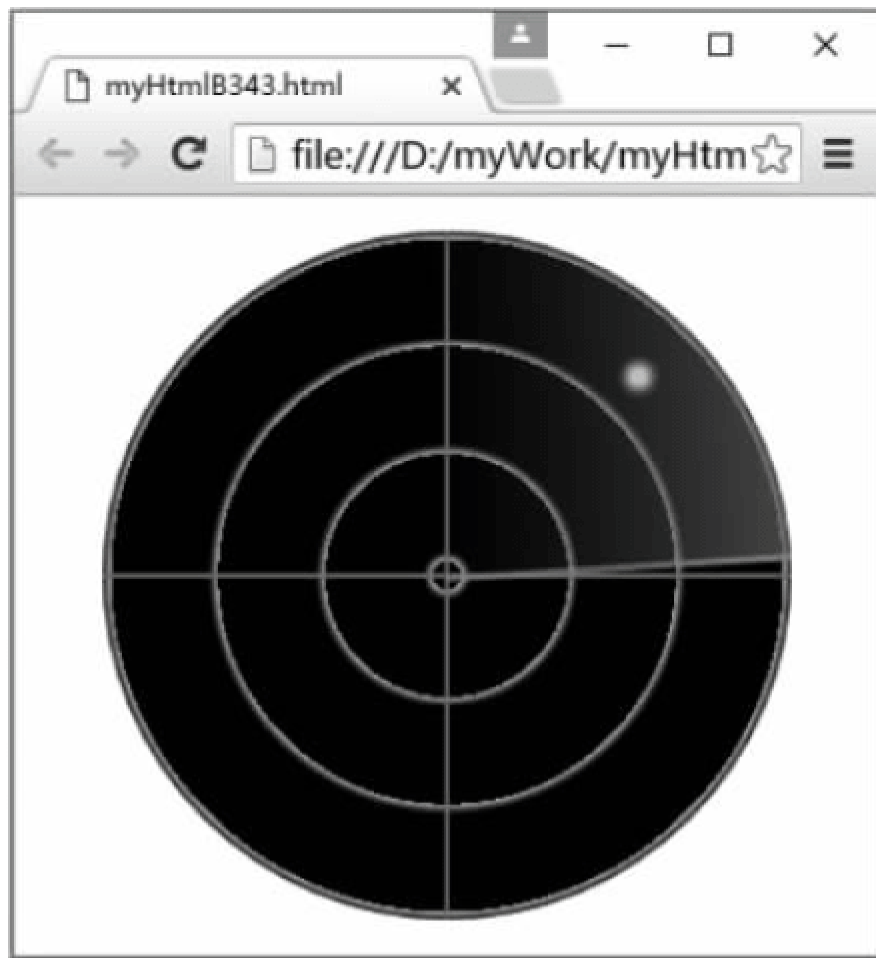


图　326-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  html, body { margin: 0; padding: 0;
             height: 100%; position: relative; overflow: hidden; }
  /*设置盒子的基本样式*/
  #myBox { width: 100%; height: 100%;
          border-radius: 50% 50% 0 0; position: relative; }
  /*设置嵌套圆环的基本样式*/
  .myCircle { position: absolute;left: 0; right: 0; top: 0; bottom: 0;
             margin: auto;width: 300px;height: 300px; border-radius: 50%; background: -webkit-
             repeating-radial-gradient(lightgreen 3%, black 5%, black 25%); }
  /*设置水平坐标线*/
  .myHorizontal { position: absolute;top: 149px; height: 2px; width: 100%;
                 background: #00A638; }
  /*设置垂直坐标线*/
  .myVertical { position: absolute;left: 149px; width: 2px;
               height: 100%; background: #00A638 }
  /*设置扫描图形*/
  .myScan { width: 150px;height: 150px; background: linear-gradient(0deg, rgba(115, 255, 162, 0.0),
          rgba(0, 166, 56, 0.75)); border-radius: 100% 0 0 0; z-index: 999; -webkit-animation:
          myScan 4s linear infinite; border-right: 3px solid rgba(0, 255, 86, 0.5); -webkit-
          transform-origin: 150px 150px; }
  /*设置闪烁点的基本样式*/
```

```
.myDot { position: absolute; border-radius: 50%; opacity: 0;
        top: 20px; left: 20px; width: 7px; height: 7px;
        box-shadow: 0 0 5px 3px lightgreen; background: #C3FFD3; }
.myDot1 { top: 75px;left: 75px; /* 设置闪烁点 1 的坐标和闪烁动画 */
         -webkit-animation: myFlash 4s linear infinite 3.5s;}
.myDot2 { top: 150px;left: 250px; /* 设置闪烁点 2 的坐标和闪烁动画 */
         -webkit-animation: myFlash 4s linear infinite 1s; }
.myDot3 { top: 250px; left: 250px; /* 设置闪烁点 3 的坐标和闪烁动画 */
         -webkit-animation: myFlash 4s linear infinite 1.5s; }
.myDot4 { top: 60px; left: 230px; /* 设置闪烁点 4 的坐标和闪烁动画 */
         -webkit-animation: myFlash 4s linear infinite 0.5s; }
/* 设置扫描关键帧 */
@-webkit-keyframes myScan { 0% { -webkit-transform: rotateZ(0deg); }
                            100% { -webkit-transform: rotateZ(360deg); } }
/* 设置闪烁关键帧 */
@-webkit-keyframes myFlash { 0% { opacity: 0; }
          10% { opacity: 1; } 30% { opacity: 0; } }
</style></head>
<body><div id="myBox"><div class="myCircle">
  <div class="myHorizontal"></div><div class="myVertical"></div>
  <div class="myScan"></div><div class="myDot myDot1"></div>
  <div class="myDot myDot2"></div><div class="myDot myDot3"></div>
  <div class="myDot myDot4"></div></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，rotateZ(0deg)和 rotateZ(360deg)表示扫描扇形将围绕 Z 轴从 0°旋转至 360°，即一圈；opacity：1 表示完全不透明，opacity：0 表示完全透明；-webkit-animation：myScan 4s linear infinite 表示在 4 秒内完成关键帧动画 myScan 中的旋转动作，且永不停歇。

此实例的源文件名是 myHtmlB343.html。

327　使用关键帧动画模拟霓虹灯文字的闪烁

此实例主要通过在关键帧动画中改变 text-shadow 属性值模拟霓虹灯文字一眨一眨的闪烁效果。当在 Google Chrome 浏览器中显示该页面时，“炫酷实例集锦”将以一眨一眨的霓虹灯效果不断闪烁，如图 327-1 所示。有关此实例的主要代码如下。



图　327-1

```
<! doctype html >< html >< head >< meta charset = "UTF - 8">
 < style type = "text/css">
  body { background - color: black; }
  # myBox { width: 400px; margin: 50px auto; text - align: center; }
  a { font - size: 16px; text - decoration: none; color: white; font - size: 60px;
      /* 设置关键帧动画的参数 */
       - webkit - animation: myNeon 0.5s ease - in - out infinite alternate; }
  /* 设置关键帧的霓虹效果 */
  @ - webkit - keyframes myNeon { 0 % ,100 % { color:white; }
      40 %  { text - shadow: 0 0 10px # FFF, 0 0 20px # FFF, 0 0 30px # FFF, 0 0 40px # FF1177,
            0 0 70px # FF1177,0 0 80px # FF1177, 0 0 100px # FF1177,0 0 150px # FF1177;}
      90 %  { text - shadow: 0 0 5px # FFF,0 0 10px # FFF, 0 0 15px # FFF,0 0 20px # FF1177,
            0 0 35px # FF1177,0 0 40px # FF1177, 0 0 50px # FF1177,0 0 75px # FF1177; } }
 </style></head>
<body><div id = "myBox" ><a>炫酷实例集锦</a></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于向文本添加一个或多个阴影（text-shadow: h-shadow v-shadow blur color;），该属性是用逗号分隔的阴影列表，每个阴影由两个或 3 个长度值和一个可选的颜色值进行规定，省略的长度是 0；-webkit-animation: myNeon 0.5s ease-in-out infinite alternate 表示在 500 毫秒内以 ease-in-out 模式完成关键帧动画 myNeon 中的文字阴影改变动作，然后交替反向，且永不停歇。

此实例的源文件名是 myHtmlB344.html。

328 使用关键帧动画实现 3D 效果的走马灯文字

此实例主要在关键帧动画中使用 translateX()方法平移文字，从而实现 3D 效果的走马灯文字滚动播放。当在 Google Chrome 浏览器中显示该页面时，“CSS3 炫酷实例集锦”将以 3D 效果滚动播放，如图 328-1 所示。有关此实例的主要代码如下。



图 328-1

```
<! doctype html >< html >< head >< meta charset = "UTF - 8">
 < style type = "text/css">
```

```
body { background: black; }
/*设置盒子的基本样式*/
#myMarquee { margin-top: 7rem; font-size: 0; -webkit-perspective: 500px;
             font-family: "Microsoft YaHei";text-align: center; }
/*设置背景的基本样式*/
#myMarquee div { display: inline-block; height: 12rem; width: 30rem;
                 position: relative; }
/*设置左背景*/
#myMarquee div:first-of-type { background: red; color: white;
 -webkit-transform-origin: top right; -webkit-transform: rotateY(-40deg); }
/*设置右背景*/
#myMarquee div:last-of-type { background: blue; color: snow;
 -webkit-transform-origin: top left; -webkit-transform: rotateY(45deg); }
/*设置文字块盒子*/
#myMarquee div { font-size: 8rem; overflow: hidden; }
/*设置文字块样式*/
#myMarquee div span { position: absolute; width: 400%; line-height: 1.4; }
/*设置左边的文字块动画*/
#myMarquee div:first-of-type span {-webkit-transform: translateX(60rem);
           -webkit-animation: myLeftcrawl 14s linear infinite;
           text-shadow: 14px 0px 14px black; }
/*设置右边的文字块动画*/
#myMarquee div:last-of-type span {-webkit-transform: translateX(30rem);
           -webkit-animation: myRightcrawl 14s linear infinite; }
/*设置左边的文字块动画关键帧*/
@-webkit-keyframes myLeftcrawl {to { -webkit-transform: translateX(-100rem);}}
/*设置右边的文字块动画关键帧*/
@-webkit-keyframes myRightcrawl {to { -webkit-transform: translateX(-130rem);}}
</style></head>
<body><div id="myMarquee">
  <div><span>CSS3 炫酷实例集锦</span></div>
  <div><span>CSS3 炫酷实例集锦</span></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-perspective：500px 用于定义 3D 元素(myMarquee)和视图的距离,以像素计,当为元素定义 perspective 属性时其子元素会获得透视效果,而不是元素本身;-webkit-transform-origin：top left 用于设置被转换元素(#myMarquee div:last-of-type)的位置;-webkit-transform：rotateY(45deg)定义沿着 Y 轴顺时针旋转 45°;-webkit-transform：translateX(60rem)表示文字块沿着 X 轴的平移尺寸;-webkit-animation：myLeftcrawl 14s linear infinite 表示在 14 秒内以 linear 模式完成关键帧动画 myLeftcrawl 中的文字走马灯动作,且永不停歇。

此实例的源文件名是 myHtmlB345.html。

329　使用关键帧动画模拟阴影跟随灯光移动

此实例主要在关键帧动画中同步改变灯光的 left 属性和文字阴影的水平阴影的位置,从而实现文字阴影跟随灯光移动的效果。当在 Google Chrome 浏览器中显示该页面时,白色的灯光将在文字的左、右两端来回移动,文字阴影也跟随灯光的移动而移动,如图 329-1 所示。有关此实例的主要代码如下。

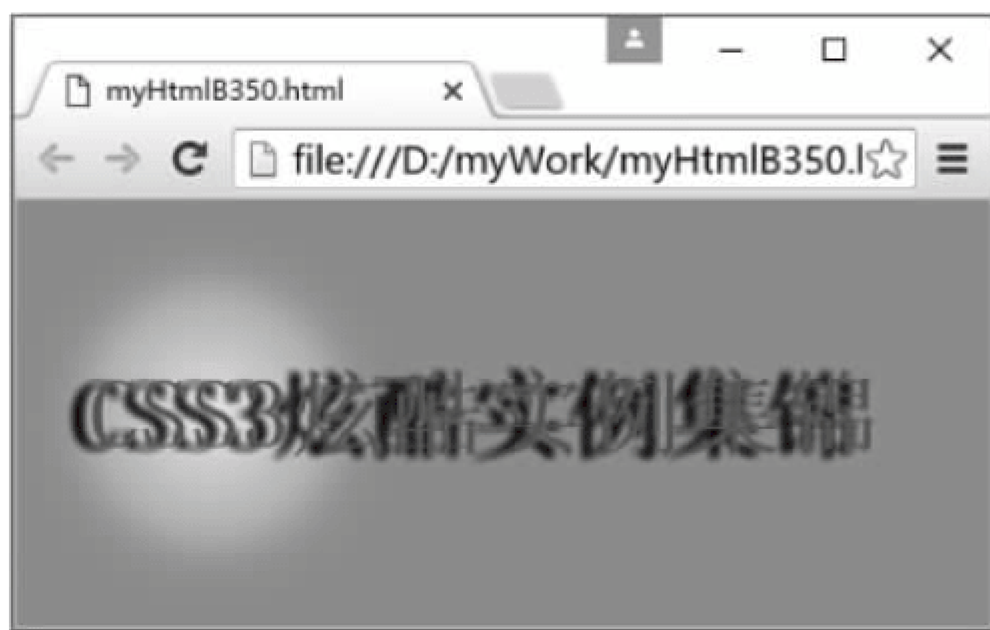


图 329-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  /* 设置背景颜色 */
  body { background: darkgray; }
  /* 设置盒子的基本样式 */
  .myBox{ position: relative;width:400px; margin: 0 auto; }
  /* 设置文字的基本样式 */
  p { margin: 0; padding: 0;font-size: 42px; position: absolute;
     top: 60px; left: 20px; color: red; font-weight: bold;z-index: 1;
     animation: myShadow 5.0s ease-in-out infinite; }
  /* 设置灯光的基本样式 */
  .myLight {z-index: 2; opacity: 0.8; position: absolute; top: 35px; left: 50px; width: 100px; height:
          100px; border-radius: 100px; z-index: 0;background: -webkit-radial-gradient(center,
          ellipse cover, #FFF 0%, rgba(255, 255, 255, 0.2) 100%);box-shadow: 0px 0px 50px #FFF;
          animation: myLamp 5.0s ease-in-out infinite; }
  /* 设置灯光关键帧样式 */
  @-webkit-keyframes myLamp{0% { left: 15px; }
      50% { left: 270px;} 100% { left: 15px; } }
  /* 设置文字阴影关键帧样式 */
  @-webkit-keyframes myShadow { 0% { text-shadow: -9px 1px 4px #000; }
    50% { text-shadow: 9px 1px 4px #000; }100% { text-shadow: -9px 1px 4px #000; } }
 </style></head>
<body><div class = "myBox"><p>CSS3 炫酷实例集锦</p>
    <div class = "myLight"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-shadow 属性用于向文本添加一个或多个阴影，其语法格式如下。

```
text-shadow: h-shadow v-shadow blur color
```

其中，参数 h-shadow 表示水平阴影的位置，允许为负值；参数 v-shadow 表示垂直阴影的位置，允许为负值；参数 blur 表示模糊的距离；参数 color 表示阴影的颜色。在此实例中主要是在关键帧动画中改变文字阴影的 h-shadow 值来同步与灯光的 left 值。

animation: myShadow 5.0s ease-in-out infinite 表示在 5 秒内以 ease-in-out 模式完成关键帧动画 myShadow 中的改变文字阴影的 h-shadow 值的动作，且永不停歇。animation: myLamp 5.0s ease-in-out infinite 表示在 5 秒内以 ease-in-out 模式完成关键帧动画 myLamp 中的改变灯光的 left 值的动作，且永不停歇。

此实例的源文件名是 myHtmlB350.html。

330　使用关键帧动画模拟多层波浪波动的效果

此实例主要在关键帧动画中不停地旋转两个圆角矩形，从而实现通过两个矩形的圆角产生类似于波浪波动的效果。当在 Google Chrome 浏览器中显示该页面时，两个圆角矩形将不停地旋转（大部分被隐藏），露出的圆角部分像波浪一样波动，如图 330-1 所示。有关此实例的主要代码如下。



图　330-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*设置盒子的基本样式*/
  .myBox { margin:3px auto; position: relative; width: 400px;height: 300px;
          /*注释下面这行代码即可看到整个圆角矩形的旋转效果*/
          overflow: hidden;transform: rotateZ(180deg); }
  /*设置产生波浪的圆角矩形的基本样式*/
  .myWave { opacity: 0.4; position: absolute; top: 3%;left: 50%;width: 500px;
          height: 500px; margin-left: -250px; margin-top: -250px;
          -webkit-transform-origin: 50% 48%; border-radius: 41%; }
  .myWave.green { background:green; /*设置绿色波浪的动画参数*/
                 -webkit-animation: myRotate 2.9s infinite linear; }
  .myWave.cyan { background:cyan; /*设置青色波浪的动画参数*/
                -webkit-animation: myRotate 4.3s infinite linear; }
  /*设置波浪的关键帧参数*/
  @-webkit-keyframes myRotate {from { -webkit-transform: rotate(360deg); } }
  /*设置提示文本的基本样式*/
  .myTitle { position: absolute;left: 0; top: 0; width: 100%; z-index: 1;
           line-height: 150px; text-align: center; font-size: 40px;
           transform: rotateZ(-180deg); }
 </style></head>
<body><div class='myBox' id="myBox">
    <div class='myWave green'></div><div class='myWave cyan'></div>
    <div class='myTitle'>CSS3 精彩实例集锦</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-radius：41%用于设置矩形的

圆角半径，它是此实例的关键，可以根据矩形的大小进行适当调整；-webkit-transform：rotate(360deg)表示将圆角矩形旋转360°；-webkit-animation：myRotate 4.3s infinite linear 表示在4300毫秒内完成关键帧动画 myRotate 中设置的将圆角矩形旋转一周的动作，并且一直旋转。注意，两个动画的时间参数不能设置成完全相同，否则就不会产生一波一波的效果。

此实例的源文件名是 myHtmlB351.html。

331 使用关键帧动画模拟生物在水中的摆动

此实例主要通过在关键帧动画中使用 translate3d()和 rotate3d()等方法模拟鱼等海洋生物在水中游动时使劲摆动身子的特效。当在 Google Chrome 浏览器中显示该页面时，类似鱼的海洋生物将使劲地摆动身子，如图 331-1 所示。有关此实例的主要代码如下。



图 331-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   function insertRule(sheet, selectorText, cssText, position) {
    if (sheet.insertRule) {
     sheet.insertRule(selectorText + "{" + cssText + "}", position);
    } else if (sheet.addRule) {
     sheet.addRule(selectorText, cssText, position);
   } }
   for (var i = 1; i <= 635; i++) {
    $(".myBox").append("<span class='myImg'></span>");
    //设置背景图像的位置
    $(".myBox .myImg:nth-child(" + i + ")").css("background-position", (-2 * i) + "px 0");
    //设置 3D 位移和旋转的初始值
    $(".myBox .myImg:nth-child(" + i + ")").css("transform", "translate3d(-50%,-40%,0) rotate3d
(0,1,0," + (360/800 * i) + "deg) translate3d(200px,0,0) rotate3d(0,-1,0," + (360/800 * i) + "deg");
    //设置关键帧
    insertRule(document.styleSheets[0], "@-webkit-keyframes ani-" + i, "50%{transform:
translate3d(-50%,-60%,0) rotate3d(0,1,0," + (360/800 * i) + "deg) translate3d(180px,0,0px)
rotate3d(0,-1,0," + (360/800 * i) + "deg);}", 0);
```

```
      //设置动画参数
      $(".myBox .myImg:nth-child(" + i + ")").css("animation", "ani-" + i + " 1s ease-in-out " + (-1/
200 * i) + "s infinite");
   }});
  </script>
  <style type="text/css">
   /* 设置盒子的基本样式 */
    .myBox { transform-style: preserve-3d;
             position: absolute;left: 45%; top: 45%;
             transform: translate3d(-50%, -50%, 0) rotate3d(1, 0, 0, 5deg); }
   /* 设置图像框的基本样式 */
    .myImg { width: 5px; height: 250px; position: absolute;
             background: url(img/B357B.png) no-repeat; }
  </style></head>
<body><div class="myBox"></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，translate3d()用于使一个元素在三维空间中移动，这种变形的特点是使用三维向量的坐标定义元素在每个方向上移动多少，其基本语法如下。

```
translate3d(tx,ty,tz)
```

其中，参数 tx 代表横向坐标位移向量的长度；参数 ty 代表纵向坐标位移向量的长度；参数 tz 代表 Z 轴位移向量的长度。

rotate3d()方法用于在 3D 空间中旋转，有 3 个维度来描述一个转动轴，轴的旋转是一个[x,y,z]向量并经过元素原点，其基本语法如下。

```
rotate3d(x,y,z,a)
```

其中，参数 x 是一个 0 到 1 的数值，主要用来描述元素围绕 X 轴旋转的矢量值；参数 y 是一个 0 到 1 的数值，主要用来描述元素围绕 Y 轴旋转的矢量值；参数 z 是一个 0 到 1 的数值，主要用来描述元素围绕 Z 轴旋转的矢量值；参数 a 是一个角度值，主要用来指定元素在 3D 空间中旋转的角度，如果其值为正数，元素顺时针旋转，反之元素逆时针旋转。

insertRule(document.styleSheets[0]," @-webkit-keyframes ani-" + i, " 50%{transform: translate3d(-50%,-60%,0) rotate3d(0,1,0,"+(360/800 * i)+"deg) translate3d(180px,0,0px) rotate3d(0,-1,0,"+(360/800 * i)+"deg);}", 0)用于设置旋转和位移的关键帧。

$(".myBox .myImg:nth-child("+i+")").css("animation", "ani-"+i+" 1s ease-in-out "+(-1/200 * i)+"s infinite")表示在 1 秒内以 ease-in-out 模式完成指定关键帧动画中的位移和旋转动作，且永不停歇。

此实例的源文件名是 myHtmlB357.html。

332 使用关键帧动画模拟碎片拼接图像特效

此实例主要通过在关键帧动画中使用 translate3d()和 rotate()等方法模拟碎片拼接图像的特效。当在 Google Chrome 浏览器中显示该页面时，可以看到多个碎片在不停地旋转和移动，如图 332-1 所示，然后逐渐拼接成一幅图像，单击“以碎片拼接方式显示图像”按钮将重新演示这一过程。有关此实例的主要代码如下。

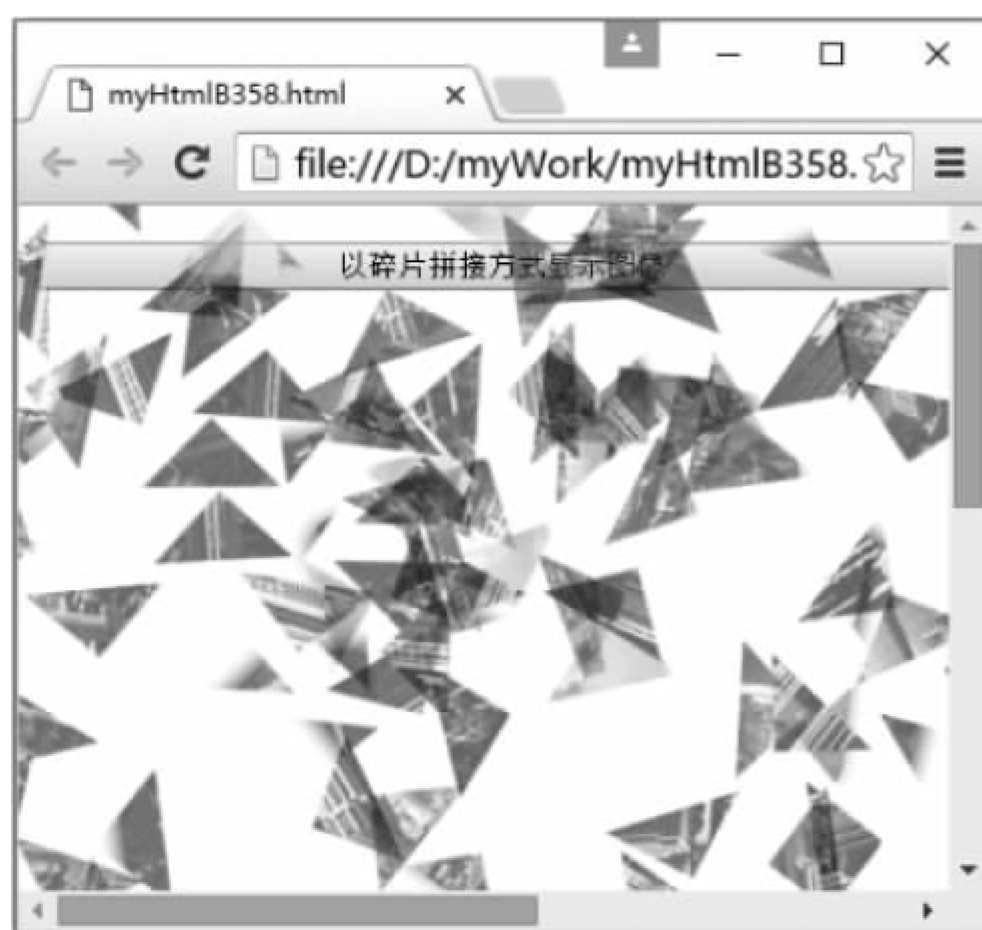


图 332-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   clipPath($("#myImage")[0]);
   $("#myBtnSplicing").click(function() {        //以碎片拼接方式显示图像
     location.reload();
  });});
 function clipPath(t) {
  t.removeAttribute("id");
  var r = {height: t.clientHeight, width: t.clientWidth, distance: 60, html: t.outerHTML};
  var a = r.distance, n = r.width, e = r.height;
  var o = "";
  for (var i = 0; i<n; i += a) {
   for (var h = 0; h<e; h += a) {
    var d = [i, h], u = [i, h + a], l = [i + a, h + a], v = [i + a, h];
    var c = [i + a/2, h + a/2];
    var m = [[d, c, v], [d, u, c], [c, u, l], [v, c, l]];
    m.forEach(function(t, a) {
     var n = t.map(function(t) {
      return t.map(function(t) {
       return t + "px"
    }).join(" ") }).join();
    //根据 n 创建路径,把图像按照路径内部的尺寸进行裁剪形成碎片
    var e = "-webkit-clip-path: polygon(" + n + ");";
    var i = Math.random();
    var h = i<.5 ? -1 : 1;
    var d = [[h * (200 + Math.round(500 * i)), -1 * (500 + Math.round(300 * i))], [h * (100 + Math.round
(500 * i)), -1 * (400 + Math.round(600 * i))], [h * (50 + Math.round(500 * i)), -1 * (500 + Math.round(300
* i))], [h * (100 + Math.round(400 * i)), 1 * (500 + Math.round(700 * i))]];
    var u = [600 * (0.5 - Math.random()), 600 * (0.5 - Math.random())];
    var l = "translate(" + u.map(function (t) {        //对碎片进行旋转和平移
     return t + "px" }).join() + ") rotate(" + Math.round(h * 360 * Math.random()) + "deg)";
    var v = "-webkit-transform:" + l + ";transform:" + l + ";";
    o = o + r.html.replace('">', '" style="' + e + v + '">')
  }) } }
  t.parentNode.innerHTML = r.html + o;
```

```
}
</script>
<style type="text/css">
  /* 控制渐变显示图像 */
  .myBox .clip:not([style]) { -webkit-animation: fadeIn 1s 5s both; }
  /* 控制拼接显示图像 */
  .myBox .clip[style] { opacity: 0; -webkit-animation: myInitial 5s both; }
  /* 控制拼接显示图像关键帧 */
  @-webkit-keyframes myInitial {
   to { opacity: 1; -webkit-transform: translate3d(0, 0, 0) rotate(0deg); } }
  /* 控制渐变显示图像关键帧 */
  @-webkit-keyframes fadeIn { from { opacity: 0; } to { opacity: 1; } }
  /* 设置图像的基本样式 */
  .image { position: absolute; width:400px;height:250px;
           border-radius: 10px;box-shadow: 2px 2px 10px black; opacity: 0; }
  button { width: 405px; }
</style></head>
<body><p><button id="myBtnSplicing">以碎片拼接方式显示图像</button></p>
<div class="myBox">
<img id="myImage" class="image clip" src="img/B358.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，translate3d()方法表示使用三维向量的坐标定义元素在每个方向上移动；rotate()方法用于根据指定的角度在2D内旋转；polygon()方法用于定义一个多边形，它的参数是一组坐标对(< shape-arg > < shape-arg >)，每一个坐标对代表多边形的一个顶点坐标，浏览器会将最后一个顶点和第一个顶点连接得到一个封闭的多边形，坐标对使用逗号进行分隔，可以使用绝对单位或百分比单位值；-webkit-animation：myInitial 5s both 表示在5秒内完成指定关键帧动画 myInitial 中的位移和旋转动作。

此实例的源文件名是 myHtmlB358.html。

333 使用关键帧动画模拟地球在太空中自转

此实例主要通过在关键帧动画中使用 translateY()和 rotateY()等方法模拟地球在太空中自转的特效。当在 Google Chrome 浏览器中显示该页面时，地球在太空中自转的效果如图 333-1 所示。有关此实例的主要代码如下。

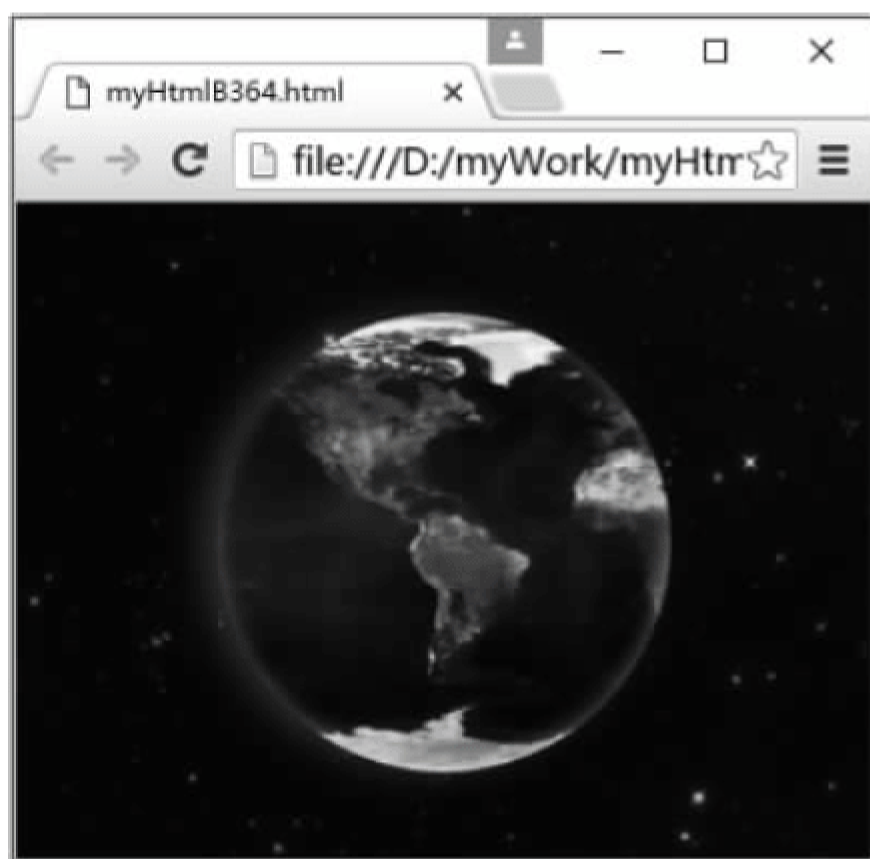


图 333-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body{ background:url("img/B364A.jpg"); }/* 设置太空图像 */
  .myEarth{ margin:0 auto; width:200px; height:200px;
           /* 设置世界地形图图像 */
           background-image:url("img/B364B.jpg");background-position: 0px 0px;
           background-size:auto 100%; border-radius: 50%;
           /* 设置动画参数 */
           -webkit-animation: myRotate 10s linear alternate infinite;
           /* 在地球左边显示外阴影、右边显示内阴影 */
           box-shadow: -8px 0 25px rgba(256,256,256,0.3), -1px -2px 14px rgba(256,256,256,0.5)
           inset; }
  /* 设置关键帧动画的参数 */
  @-webkit-keyframes myRotate {
           0%{background-position-x:0px; -webkit-transform: translateY(20%) rotateY(0deg);}
           25%{background-position-x:300px; -webkit-transform: translateY(20%) rotateY(20deg);}
           75%{background-position-x:600px; -webkit-transform: translateY(20%) rotateY(-20deg);}
           100%{background-position-x:100%; -webkit-transform: translateY(20%) rotateY(0deg);} }
 </style></head>
<body><div class="myEarth"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：-8px 0 25px rgba(256，256，256，0.3)，－1px －2px 14px rgba(256，256，256，0.5) inset 中的 inset 表示内置阴影，即阴影产生在元素的边框内部(此实例即左边)，默认情况下阴影产生在元素的边框外部(此实例即右边)；background-position-x 表示背景图像的水平坐标值。translateY()用于设置元素在 Y 轴上平移的距离；rotateY()用于设置元素在 Y 轴上旋转的角度；-webkit-animation：myRotate 10s linear alternate infinite 表示在 10 秒内以线性模式完成指定关键帧动画 myRotate 中的位移和旋转动作，且交替进行，永不停止。

此实例的源文件名是 myHtmlB364.html。

334　使用关键帧动画模拟文字弹跳滑入页面

此实例主要通过在关键帧动画中使用 translate3d()等方法模拟文字从屏幕之外弹跳滑入页面的特效。当在 Google Chrome 浏览器中显示该页面时，单击“文字从左侧弹跳进入页面”按钮，则文字将从屏幕左侧滑入页面，并在目标位置显示类似于篮球触地的弹跳特效，如图 334-1 所示；单击“文字从右侧弹跳进入页面”按钮，则文字将从屏幕右侧滑入页面，并在目标位置显示类似于篮球触地的弹跳特效。有关此实例的主要代码如下。



图　334-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnleft").click(function() {          //文字从左侧弹跳进入页面
    $('#myBox').removeClass().addClass('myFromLeft animated').one('webkitAnimationEnd', function() {
                 $(this).removeClass(); });});
   $("#myBtnright").click(function() {         //文字从右侧弹跳进入页面
    $('#myBox').removeClass().addClass('myFromRight animated').one('webkitAnimationEnd', function() {
                 $(this).removeClass(); });});});
 </script>
 <style type="text/css">
  /*设置动画参数*/
  .animated { -webkit-animation-duration: 1s; -webkit-animation-fill-mode: both; }
   /*设置左侧的弹跳动画*/
   .myFromLeft { -webkit-animation-name: myFromLeft; }
   @-webkit-keyframes myFromLeft {
              0%, 100%, 60%, 75%, 90% { -webkit-transition-timing-function: cubic-bezier
(0.215, 0.61, 0.355, 1); }
              0% {opacity: 0; -webkit-transform: translate3d(-3000px, 0, 0);}
              60% {opacity: 1; -webkit-transform: translate3d(25px, 0, 0);}
              75% {-webkit-transform: translate3d(-10px, 0, 0);}
              90% {-webkit-transform: translate3d(5px, 0, 0);}
              100% {-webkit-transform: none;} }
  /*设置右侧的弹跳动画*/
  .myFromRight { -webkit-animation-name: myFromRight; }
    @-webkit-keyframes myFromRight { 0%, 100%, 60%, 75%, 90% { -webkit-transition-timing-
function: cubic-bezier(0.215, 0.61, 0.355, 1); }
              0% { opacity: 0; -webkit-transform: translate3d(3000px, 0, 0); }
              60% { opacity: 1; -webkit-transform: translate3d(-25px, 0, 0);}
              75% { -webkit-transform: translate3d(10px, 0, 0); }
              90% { -webkit-transform: translate3d(-5px, 0, 0); }
              100% { -webkit-transform: none; } }
  /*设置文字的基本样式*/
  h1 { background-image: -webkit-linear-gradient(92deg, green, red);
       -webkit-background-clip: text; -webkit-text-fill-color: transparent;
       font-size: 3rem; font-weight: bold;   }
  #myBox { display: block; }                    /*设置盒子的显示模式*/
  button { margin: 10px; width: 180px;}
 </style></head>
<body><div align="center">
    <button id="myBtnleft">文字从左侧弹跳进入页面</button>
    <button id="myBtnright">文字从右侧弹跳进入页面</button>
    <span id="myBox"><h1>CSS3 炫酷实例集锦</h1></span></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-transform：translate3d(—3000px，0，0)表示平移动作定位在X轴的负方向的3000px位置点，这个位置点通常在大多数计算机屏幕之外的左侧；-webkit-animation-name：myFromLeft表示为动画指定@keyframes属性指定的名称，在使用这一属性后必须规定animation-duration属性，否则由于动画的时长为0，就不会播放动画了；$('#myBox').removeClass().addClass('myFromLeft animated').one('webkitAnimationEnd'，function(){ $(this).removeClass()；})是一段jQuery代码，它表示首先移除文字盒子的所有CSS类，然后添加动画类myFromLeft和animated。

此实例的源文件名是myHtmlB368.html。

335 通过关键帧动画使字符串呈现波浪抖动

此实例主要在关键帧动画中使用 translateY()方法改变字符在 Y 轴上的位置，从而使字符串呈现波浪式抖动的特效。当在 Google Chrome 浏览器中显示该页面时，字符串呈现波浪式抖动的效果如图 335-1 所示。有关此实例的主要代码如下。



图 335-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   function wave() {//拆分字符串并设置每个字符的样式
    var myString = $(".myBook")[0].textContent.trim();
    $(".myBook")[0].innerHTML = null;
    split_myString( $(".myBook"), myString);
    function split_myString(element, myArray) {
     for (var i = 0; i < myArray.length; i++) { element.append("<span>" + (myArray[i] === " " ? " " :
myArray[i]) + "</span>"); }  }
   for (var i = 0; i < 200; i++) { $(".myBook :nth-child(" + i + ")").css("animation-delay", (-0.16 +
0.016 * i) + "s");  } }
  wave();
 });
 </script>
 <style type="text/css">
  /*设置文本的基本样式*/
  .myBook span { font-size: 24px; font-weight: bold; display: inline-block; margin-top: 20px; letter
               -spacing: 2px; color: darkgreen; animation: myWave 0.08s infinite alternate; }
  /*设置动画的关键帧参数*/
  @keyframes myWave { from { transform: translateY(0); }
                     to { transform: translateY(18px); } }
 </style></head>
<body><div align="center"><div class="myBook"> HTML5 + CSS3 炫酷实例集锦</div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，.myBook :nth-child 中的 myBook 和:nth-child 之间有个空格，且必须存在，nth-child(n)选择器用于匹配属于其父元素的第 n 个子元素，而不论元素的类型，n 可以是数字、关键字或公式；translateY()方法用于根据参数指定的距离在 Y 轴上平移元素；animation: myWave 0.08s infinite alternate 表示在 80 毫秒内完成 myWave 关键帧指定的 Y 轴平移动作，并且重复、交替进行，永不停止。

此实例的源文件名是 myHtmlB372.html。

336 使用 requestAnimFrame()方法动态绘制火苗

此实例主要实现了使用 requestAnimFrame()方法创建一束不断燃烧的火苗。当在 Google Chrome 浏览器中显示该页面时将显示一束不断燃烧的火苗，如图 336-1 所示。有关此实例的主要代码如下。



图 336-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script>
  $(document).ready(function() {
   var mySurface=document.getElementById("mySurface").getContext("2d");
   var myParticles=[];                    //设置帧集合 myParticles
   var myCount=150;                       //任意改变此值,会出现奇异的火苗
   for (var i=0; i<myCount; i++) {
    myParticles.push(new particle());
   }
   var myWidth=320;var myHeight=480;      //设置画布的大小
   $("#mySurface").css({width: myWidth, height: myHeight});
   window.requestAnimFrame=(function() {  //设置动画帧
     return window.requestAnimationFrame ||
           window.webkitRequestAnimationFrame ||
           window.mozRequestAnimationFrame ||
          function (callback) {           //定义每秒执行 60 次动画
             window.setTimeout(callback, 1000/60);
   }; })();
   function particle() {
```

```
        this.speed = {x: -1 + Math.random() * 2, y: -5 + Math.random() * 5};
        myWidth = (document.getElementById("mySurface").width);
        myHeight = (document.getElementById("mySurface").height);
        this.location = {x: myWidth/2, y: (myHeight/2) + 35};
        this.radius = 0.5 + Math.random() * 1;
        this.life = 10 + Math.random() * 10;
        this.death = this.life;
        this.r = 255;
        this.g = Math.round(Math.random() * 155);
        this.b = 0;
      }
      function ParticleAnimation() {
       mySurface.globalCompositeOperation = "source-over";
       mySurface.fillStyle = "black";
       mySurface.fillRect(0, 0, myWidth, myHeight);
       mySurface.globalCompositeOperation = "lighter";
       for (var i = 0; i < myParticles.length; i++) {
        var myFrame = myParticles[i];
        mySurface.beginPath();
        myFrame.opacity = Math.round(myFrame.death/myFrame.life * 100)/100
        var gradient = mySurface.createRadialGradient(myFrame.location.x,
                              myFrame.location.y,0,myFrame.location.x,
                              myFrame.location.y,myFrame.radius);
        gradient.addColorStop(0, "rgba(" + myFrame.r + ", " + myFrame.g +
                                   ", " + myFrame.b + ", " + myFrame.opacity + ")");
        gradient.addColorStop(0.5, "rgba(" + myFrame.r + ", " + myFrame.g +
                                   ", " + myFrame.b + ", " + myFrame.opacity + ")");
        gradient.addColorStop(1, "rgba(" + myFrame.r + ", " + myFrame.g +
                                    ", " + myFrame.b + ", 0)");
        mySurface.fillStyle = gradient;
        mySurface.arc(myFrame.location.x, myFrame.location.y,
                                             myFrame.radius, Math.PI * 2, false);
        mySurface.fill(); myFrame.death--; myFrame.radius++;
        myFrame.location.x += (myFrame.speed.x);
        myFrame.location.y += (myFrame.speed.y);
        //重新生成 myParticles
        if (myFrame.death < 0||myFrame.radius < 0) {
           myParticles[i] = new particle();
       } }
       //通过 window.requestAnimFrame()方法递归调用自身
       window.requestAnimFrame(ParticleAnimation);
      }
      ParticleAnimation();              //开始动画
     });
    </script></head>
   <body><div><canvas id = "mySurface"></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，window.requestAnimFrame()方法是 HTML5 新增的方法，该方法用于以一种比定时器更好的性能来实现动画。通过 window.requestAnimFrame()方法，浏览器可以将各种并发性动画结合到一个单一的页面进行创建及渲染，这种能力将使动画的实现具有更好的性能。例如，可以同时执行使用 JavaScript 脚本代码实现的动画与使用 CSS 中的 transition 样式属性实现的动画。另外，通过 window.requestAnimFrame()方法，当

用户将浏览器标签窗口切换到其他标签窗口时，当前页面中的动画将被暂停运行，以减少 CPU、GPU 与内存的消耗。

此实例的源文件名是 myHtmlA107.html。

337 使用 requestAnimFrame()方法动态绘制桃心

此实例主要通过使用 requestAnimFrame()方法以动画的形式动态绘制桃心图形的线条。当在 Google Chrome 浏览器中显示该页面时，动态绘制桃心图形的过程如图 337-1 所示。有关此实例的主要代码如下。

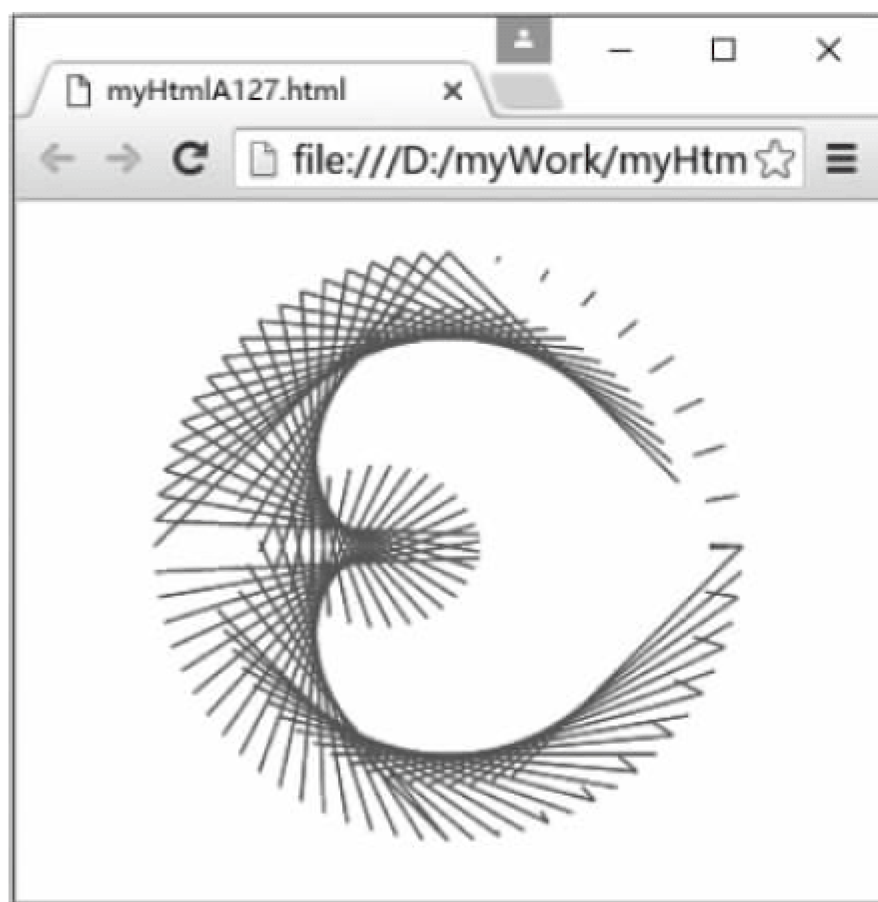


图 337-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   myCanvas = document.getElementById("myCanvas");
   myContext = myCanvas.getContext('2d');
   W = myCanvas.width = window.innerWidth - 20;        //设置画布的宽度
   H = myCanvas.height = window.innerHeight - 20;      //设置画布的高度
   radius = H * 0.45;                                  //设置半径的大小
   ticks = [];
   for (var i = 0; i <= 360; i += 5) { ticks.push(i);  //向数组的末尾添加 i
   }
   function toRadians(angle) {
     return angle * (Math.PI/180);                     //将角度转换成弧度
   }
   function calcPoints() {
    points = [];
    for (var i = 0; i < ticks.length; i++) {
     points.push([ W/2 + Math.cos(toRadians(ticks[i])) * radius,
       H/2 + Math.sin(toRadians(ticks[i])) * radius ]);
    }; };
   function drawHeart(dashL, dash0) {                  //绘制桃心图案
    myContext.beginPath();                             //开始绘制路径
    myContext.lineWidth = 1.2;                         //线条的宽度
```

```
        myContext.setLineDash([dashL, dash0]);            //设置线条样式
        myContext.strokeStyle = 'red';                    //线条的颜色
        offset = 18;
        for (var i = 0; i <= 18; i++) {
         myContext.moveTo(points[i][0], points[i][1]);
         myContext.lineTo(points[i + offset][0], points[i + offset][1]);     }
        for (var i = 36; i <= 54; i++) {
         myContext.moveTo(points[i][0], points[i][1]);
         myContext.lineTo(points[i + offset][0], points[i + offset][1]);     }
        offset = 20;
        for (var i = 18; i < 36; i++) {
         myContext.moveTo(points[i][0], points[i][1]);
         myContext.lineTo(points[i + offset][0], points[i + offset][1]);
         offset++;     }
        offset = 20;
        for (var i = 54; i > 36; i -- ) {
         myContext.moveTo(points[i][0], points[i][1]);
         myContext.lineTo(points[i - offset][0], points[i - offset][1]);
         offset++;
        }
         myContext.stroke();
        myContext.closePath();
       }
       calcPoints(); update();
       var dashLength = 100; var dashOffset = 100;
       function update() {
        if (dashLength < 650) {
         dashLength += 2; myContext.clearRect(0, 0, W, H);
         drawHeart(dashLength, dashOffset);
        }
        window.requestAnimationFrame(update);     //请求动画不断绘制心形线条
       } });
     </script></head>
    <body><div align = "center"><canvas id = "myCanvas"></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，window. requestAnimationFrame(update)用于通过传入的 update()方法递归调用自己绘制桃心。requestAnimationFrame()方法的语法格式如下。

```
requestID = window.requestAnimationFrame(callback);
```

在每次需要重新绘制动画时，调用 callback 参数所指定的函数。此时回调函数会收到一个参数，这个 DOMHighResTimeStamp 类型的参数指示当前时间距开始触发 requestAnimationFrame 的回调时间。该方法的 requestID 是一个长整型非零值，作为一个唯一的标识符，可以将该值作为参数传给 window. cancelAnimationFrame()来取消这个回调函数。

此实例的源文件名是 myHtmlA127. html。

338 通过矩阵变换绘制动态走动的时钟

此实例主要通过使用 transform()方法进行矩阵变换，从而绘制动态走动的时钟。当在浏览器中显示该页面时将显示一个带刻度盘的不停走动的时钟，如图 338-1 所示。有关此实例的主要代码如下。



图 338-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script language = "javascript">
  window.onload = function() {
                               myClock();
  };
  function myClock() {
   var myCanvas = document.getElementById("myCanvas");
   var myContext = myCanvas.getContext("2d");
   myContext.clearRect(0, 0, 410, 410);
   myContext.save();                            //保存状态
   myContext.lineWidth = 2;                     //绘制刻度
   myContext.fillStyle = "red";
   myContext.strokeStyle = "black";
   var sin = Math.sin(Math.PI/6);
   var cos = Math.cos(Math.PI/6);
   myContext.translate(205, 205);
   for (var i = 0; i <= 12; i++) {
    myContext.fillRect(160, -7.5, 30, 10);
    myContext.strokeRect(160, -7.5, 30, 10);
    myContext.transform(cos, sin, -sin, cos, 0, 0);
   }
   var sin = Math.sin(Math.PI/30);              //重置中心位置
   var cos = Math.cos(Math.PI/30);
   for (var i = 0; i <= 60; i++) {
    myContext.fillRect(170, -5, 10, 2);
    myContext.transform(cos, sin, -sin, cos, 0, 0);
   }
   myContext.restore();
   myContext.font = '14pt 宋体';
```

```
        myContext.fillText("12", 193, 62);                          //绘制显示的4个时钟数字
        myContext.fillText("6", 202, 360);
        myContext.fillText("9", 50, 215);
        myContext.fillText("3", 350, 210);
        myContext.save();
        myContext.translate(205, 205);
        myContext.save();
        var now = new Date();                                       //获取当前时间
        var hrs = now.getHours();
        var min = now.getMinutes();
        var sec = now.getSeconds();
        myContext.rotate(Math.PI/6 * (hrs + (min/60) + (sec/3600)));  //绘制时针
        myContext.beginPath();
        myContext.lineWidth = 6;
        myContext.moveTo(0, 10);
        myContext.lineTo(0, -60);
        myContext.stroke();
        myContext.restore();
        myContext.save();
        myContext.rotate(Math.PI/30 * (min + (sec/60)));           //绘制分针
        myContext.beginPath();
        myContext.lineWidth = 4;
        myContext.moveTo(0, 20);
        myContext.lineTo(0, -110);
        myContext.stroke();
        myContext.restore();
        myContext.save();
        myContext.rotate(Math.PI/30 * sec);                         //绘制秒针
        myContext.strokeStyle = "#E33";
        myContext.beginPath();
        myContext.lineWidth = 2;
        myContext.moveTo(0, 20);
        myContext.lineTo(0, -140);
        myContext.stroke();
        myContext.restore();                                        //恢复状态
        myContext.restore();                                        //必须两次
        setTimeout(myClock, 1000);                                  //每秒刷新一次
      }
    </script></head>
   <body><canvas id = "myCanvas" width = "410" height = "410"></canvas>
   </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,transform()方法允许缩放、旋转、移动并倾斜当前的环境,该变换只会影响调用 transform()方法之后的绘图。如果已经将绘图设置为放大两倍,transform()方法把绘图放大两倍,绘图最终将放大4倍。transform()方法的语法声明如下。

```
transform(a,b,c,d,e,f);
```

其中,参数a表示水平缩放绘图;参数b表示水平倾斜绘图;参数c表示垂直倾斜绘图;参数d表示垂直缩放绘图;参数e表示水平移动绘图;参数f表示垂直移动绘图。

此实例的源文件名是 myHtmlA066.html。

339　通过绘制圆弧模拟风车叶轮的转动特效

此实例主要通过使用 arc()方法和 setInterval()函数以绘制圆弧的形式模拟风车叶轮的转动特效。当在 Google Chrome 浏览器中显示该页面时，风车的 4 个叶轮将连续不断地围绕圆心转动，如果在“风车转动速度:”下拉列表框中选择风速，再单击“设置风速”按钮，则风车的 4 个叶轮将按照选择的风速进行旋转，如图 339-1 所示。有关此实例的主要代码如下。



图　339-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  var canvas,context; var X,Y;                //风车的圆心坐标
  var rate=1;                                 //风车的转速,默认为 1
  var R=20;                                   //风车的半径
  var canvasWidth,canvasHeight;
  function drawArc(radian, style){
   context.beginPath();
   var myGradient=context.createLinearGradient(X, Y, X, Y+4*R);
   myGradient.addColorStop(0, style);
   myGradient.addColorStop(1, "white");
   context.fillStyle=myGradient;
   context.arc(X-(2*R*Math.cos(radian)),
             Y-2*R*Math.sin(radian), 2*R, radian, Math.PI+radian, true);
   context.closePath();context.fill(); context.save();
  }
  var radian=0;
  function draw(){
   radian=radian+(rate*1/32)*Math.PI;
```

```
        var myGradient = context.createLinearGradient(X, Y, X + 10, Y);
        myGradient.addColorStop(0, '#663300');
        myGradient.addColorStop(0.4, '#996600');
        myGradient.addColorStop(1, '#552200');
        context.fillStyle = myGradient;
        context.clearRect(0,0,canvasWidth,canvasHeight);
        context.fillRect(X, Y, 0.2 * R, 8 * R);
        drawArc(radian,"#552200");                    //绘制4个叶轮
        drawArc(radian + 1/2 * Math.PI, "#990000");
        drawArc(radian + Math.PI, "#0033FF");
        drawArc(radian + 3/2 * Math.PI, "#225500");
       }
       $(function(){
        context = $("#myCanvas")[0].getContext("2d");
        canvasWidth = $("#myCanvas")[0].width;canvasHeight = $("#myCanvas")[0].height;
        X = canvasWidth/2; Y = canvasHeight/3;
        //每隔20毫秒执行一次绘制操作
        setInterval(draw,20);
        //响应单击"设置风速"按钮
         $("#myBtnRate").click(function(){ rate = $("select").val(); });
       });
     </script></head>
    <body><div align = "center">
     <canvas id = "myCanvas" height = "300px" width = "400px" ></canvas><br>
     <p>风车转动速度:
      <select><option value = "0">停止</option><option value = "0.1">微风</option>
              <option value = "0.5">小风</option><option value = "1">中风</option>
             <option value = "2">大风</option><option value = "4">狂风</option></select>
      <button id = "myBtnRate">设置风速</button></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，arc()方法用于创建弧/曲线，或者用于创建圆、部分圆。如果需要通过 arc()创建圆，应把起始角设置为 0、结束角设置为 2 * Math.PI。arc()方法的语法声明如下。

```
arc(x,y,r,sAngle,eAngle,counterclockwise)
```

其中，参数 x 表示圆心的 x 坐标；参数 y 表示圆心的 y 坐标；参数 r 表示圆的半径；参数 sAngle 表示起始角，以弧度计，弧(圆形)的三点(时)钟位置是 0°；参数 eAngle 表示结束角，以弧度计；参数 counterclockwise 是可选参数，规定是逆时针还是顺时针绘图，false 表示顺时针，true 表示逆时针。

setInterval()函数用于按照指定的周期来调用函数或计算表达式，该方法会不停地调用参数中指定的方法，直到 clearInterval()函数被调用或窗口被关闭。由 setInterval()函数返回的 ID 值可用作 clearInterval()函数的参数。setInterval()函数的语法格式如下。

```
setInterval(code,millisec[,"lang"])
```

其中，参数 code 表示要调用的方法或要执行的代码串；参数 millisec 表示周期性执行或调用 code 之间的时间间隔，以毫秒计。该方法的返回值用于传递给 clearInterval()函数从而取消对 code 的周期性执行的值。

此实例的源文件名是 myHtmlA119.html。

340　通过旋转绘图对象模拟高速旋转的车轮

此实例主要使用画布的 translate()、rotate()、drawImage()方法和定时器方法 setInterval()，从而实现以旋转绘图对象(画布)模拟高速旋转的车轮。当在 Google Chrome 浏览器中显示该页面时，单击“开始转动”按钮，则车轮将高速转动，如图 340-1 所示；单击“停止转动”按钮，则车轮将停止转动。有关此实例的主要代码如下。



图　340-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var canvas = $("#myCanvas")[0];
   var context = canvas.getContext("2d");
   var angle = 0, timer;
   $("#myBtnStart").click(function() {                //响应鼠标单击"开始转动"按钮
    timer = setInterval(function() {                  //每隔 50 秒旋转 1°
     if (angle == 360) { angle = 0; }
     context.clearRect(0, 0, 200, 200);
     context.translate(100, 100);
     context.rotate(angle * Math.PI/180);             //根据指定的角度(弧度)旋转绘图
     context.translate(-100, -100);
     context.drawImage($("img")[0], 0, 0, 200, 200);  //绘制图像
     angle++; }, 50); });
  $("#myBtnStop").click(function() {                  //响应鼠标单击"停止转动"按钮
    clearInterval(timer);
  });
  $("#myBtnStart").click();                           //自动执行鼠标单击"开始转动"按钮的动作
 });
 </script>
 <style type="text/css">
  img { display: none; }
  button { width: 150px; }
 </style></head>
<body><div align="center"><img src="img/A120.jpg"/>
  <canvas id="myCanvas" width="200" height="200"></canvas>
```

```
<p><button id="myBtnStart">开始转动</button>
   <button id="myBtnStop">停止转动</button></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，context.clearRect(0, 0, 200, 200)用于清空(0, 0, 200, 200)参数指定的矩形内的所有像素；context.translate(100, 100)用于根据指定参数(100,100)的位置重新映射画布上的(0,0)位置；context. rotate(angle * Math.PI/180)用于以指定的角度(弧度)旋转当前的绘图(画布)；context.drawImage($("img")[0], 0, 0, 200, 200)用于在画布上根据指定位置和尺寸绘制(车轮)图像；setInterval()函数用于按照指定的周期(50毫秒)执行旋转画布对象及绘图的代码，该方法会不停地调用参数中指定的代码，直到clearInterval()函数被调用或窗口被关闭。

此实例的源文件名是myHtmlA120.html。

341 通过定时器改变HSLA值模拟渐变的圆环

此实例主要使用不断改变的HSLA颜色填充圆环，从而使圆环的颜色在圆周上呈现渐变的效果。当在Google Chrome浏览器中显示该页面时，圆环的颜色将不断动态改变，如图341-1所示。有关此实例的主要代码如下。



图 341-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var myContext = $("canvas")[0].getContext('2d'), myHue = 0, myAngle = 0.01;
   setInterval(function() {     //每隔5毫秒更改一次颜色
    myAngle += 0.03; myHue <= 360 ? myHue += 0.25 : myHue = 0;
    var s = - Math.sin(myAngle); var c = Math.cos(myAngle);
    myContext.save(); myContext.globalAlpha = 0.5;
    myContext.beginPath();
    myContext.fillStyle = 'hsla(' + myHue + ',100%,50%,1)';   //填充绘画的颜色
    //创建弧/曲线(创建圆或部分圆)
    myContext.arc(200/2 + (s * 75), 200/2 + (c * 75), 5, 0, 2 * Math.PI);
    myContext.fill(); myContext.restore();   }, 5);
   $("canvas")[0].width = 200; $("canvas")[0].height = 200;
 });
 </script></head>
<body><div align="center"><canvas></canvas></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myContext.fillStyle='hsla('+myHue+',100%,50%,1)'用于以 HSLA 颜色的形式设置填充颜色。在 HTML5 中除了可以使用 RGB 颜色外，还可以使用 HSLA 颜色。HSLA 是使用色调(H)、饱和度(S)、亮度(L)、alpha 通道值(A)来定义颜色。HSLA 颜色标记法的意义如下。

(1) H：Hue(色调)，0(或 360)表示红色，120 表示绿色，240 表示蓝色，也可取其他数值来指定颜色，其取值范围为 0～360。

(2) S：Saturation(饱和度)，其取值范围为 0.0%～100.0%。

(3) L：Lightness(亮度)，其取值范围为 0.0%～100.0%。

(4) A：Alpha 透明度，其取值范围为 0～1。

当色调值大于 360 时(例如 480)，则实际值等于 480 除去 360 之后的余数 120(取模运算)。

此实例的源文件名是 myHtmlA178.html。

342 使用蒙版高仿电波逐级扩散的动态特效

此实例主要在-webkit-mask 蒙版上创建渐变色的圆环，从而实现高仿电波逐级扩散的动态特效。当在 Google Chrome 浏览器中显示该页面时，单击图像上的任意一点，即以此点为中心产生一个渐变色的圆环并向外扩散，连续单击此点，则会叠加这些渐变色的圆环并向外扩散，犹如基站发射电波信号一样，效果如图 342-1 所示。有关此实例的主要代码如下。

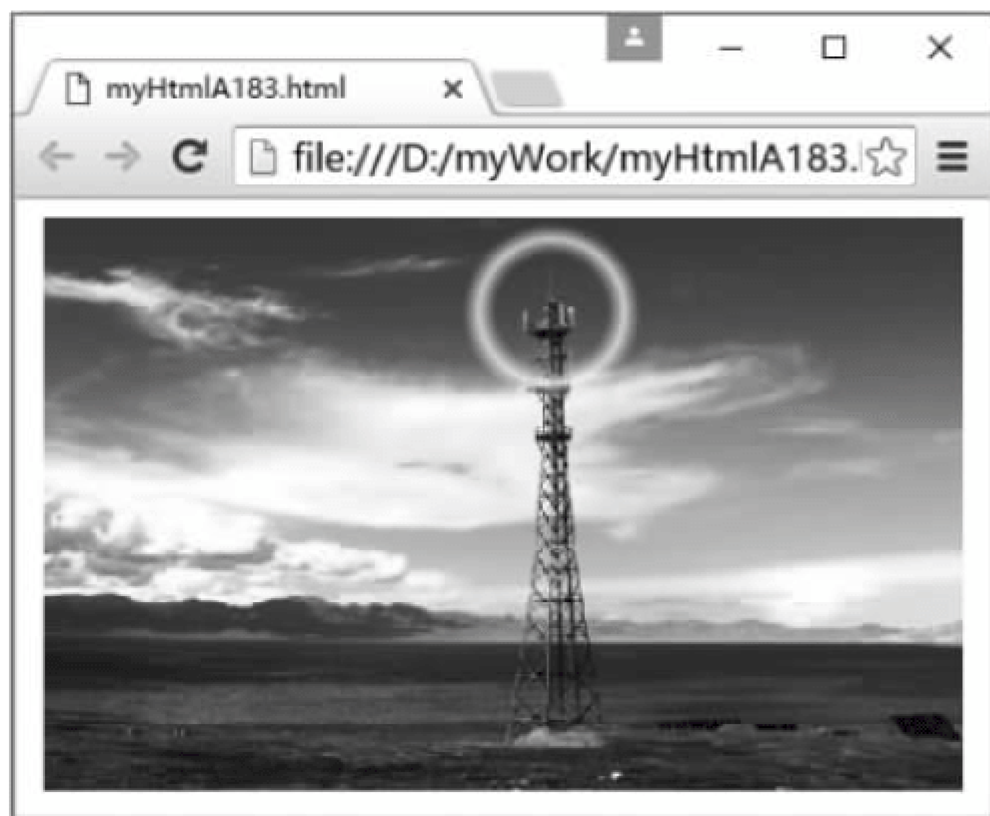


图 342-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
 //单击图片即可实现电波扩散特效，在同一位置多次单击则可实现多个波纹叠加的特效
  $("img").click(function(e) {
   var radius=0;
   var timer=setInterval(function() {
    //超出最大半径时重新从圆心扩散
    if (radius>parseInt($("img").height()/2)) { radius=0;  }
    //在蒙版上添加渐变色圆环
     $("img").css({"-webkit-mask": "-webkit-gradient(radial, "+e.offsetX+" "+e.offsetY+","+
radius+", "+e.offsetX+" "+e.offsetY+", "+(radius+15)+", from(rgba(255, 255, 255,1)), color-
stop(0.5,rgba(255, 255, 255, 0.2)), to(rgba(255, 255, 255,1)))"});
```

```
        radius += 3; }, 50);
    }) });
   </script></head>
<body><div align = "center"><img src = "img/A183.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("img").css({"-webkit- mask": "-webkit-gradient(radial, "+e.offsetX+" "+e.offsetY+","+radius+", "+e.offsetX + " "+e.offsetY+", "+(radius+15)+", from(rgba(255, 255, 255,1)), color-stop(0.5,rgba(255, 255, 255, 0.2)), to(rgba(255, 255, 255,1)))"})用于在蒙版上添加渐变色圆环。-webkit-mask 使 Google Chrome 浏览器支持创建任意形状的花样蒙版成为可能，蒙版可以是 CSS3 渐变或者半透明的 png 图片。当蒙版元素的 alpha 值为 0 的时候会覆盖下面的元素，为 1 的时候会完全显示下面的内容。与-webkit-mask 相关的属性有-webkit-mask-clip、-webkit-mask-position 和-webkit-mask-repeat 等，严重依赖来自 background 中的语法。setInterval()函数用来对遮罩层的渐变位置进行动态变化。

此实例的源文件名是 myHtmlA183.html。

343　以动画形式模拟订单提交前的等待状态

此实例主要使用@keyframes 规则创建每个关键帧动画，从而实现以动画形式模拟订单提交等进程等待状态。当在 Google Chrome 浏览器中显示该页面时，在第 1 秒显示"订单正在提交中."，如图 343-1 所示，以此类推。如果完成全部 10 秒动作，则重新开始循环。有关此实例的主要代码如下。



图　343-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  dot { display: inline - block; height: 1em; line - height: 1;
        text - align: left; vertical - align:  - 0.25em; overflow: hidden;}
  dot::before {display: block; animation: dot 10s infinite step - start both;
               content: '.........\A........\A.......\A......
               \A.....\A....\A...\A..\A.\A.';
                              white - space: pre - wrap;     /* 也可以是 white - space: pre */}
```

```
    @keyframes dot { 10 % { transform: translateY( - 9em); }
     20 % { transform: translateY( - 8em); } 30 % { transform: translateY( - 7em); }
     40 % { transform: translateY( - 6em); } 50 % { transform: translateY( - 5em); }
     60 % { transform: translateY( - 4em); } 70 % { transform: translateY( - 3em); }
     80 % { transform: translateY( - 2em); } 90 % { transform: translateY( - 1em); } }
    img{ width:487px; height:323px; border - radius: 5px; }
    body{ font - size:20px; font - weight: 500; }
  </style></head>
<body><img src = "img/B122.jpg"/>订单正在提交中<dot>.........</dot></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,@keyframes 用于设置在整个动画过程中每个关键帧的 CSS 样式,在动画过程中能够多次改变这套 CSS 样式,可以通过百分比来改变发生的时间,或者通过关键字 from 和 to,等价于 0%和 100%,其中 0%是动画的开始时间、100%动画的结束时间。@keyframes 的语法格式如下。

```
@keyframes animationname {keyframes - selector {css - styles;}}
```

其中,各参数值的说明如下。

(1) animationname:定义动画的名称。

(2) keyframes-selector:设置动画时长的百分比,合法的值有:0%～100%、from(与 0%相同)、to(与 100%相同)。

(3) css-styles:定义一个或多个合法的 CSS 样式属性。

此实例的源文件名是 myHtmlB122.html。

344 以文本缩进形式模拟等待的变长省略号

此实例主要在@keyframes 中使用 text-indent 属性规定指定的关键帧的文本缩进值,从而实现以动画形式模拟订单提交等进程等待状态的变长省略号。当在 Google Chrome 浏览器中显示该页面时,在第 1 秒显示“订单正在提交中.”,如图 344-1 所示,以此类推。如果完成全部 10 秒动作,则重新开始循环。有关此实例的主要代码如下。



图 344-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  dot { display: inline-block; width: 10ch;vertical-align: bottom;
        overflow: hidden; animation: dot 10s infinite step-start both;
        font-family: Consolas, Monaco, monospace; }
  @keyframes dot { 10% {text-indent: -9ch;}
                   20% {text-indent: -8ch;}
                   30% {text-indent: -7ch;}
                   40% {text-indent: -6ch;}
                   50% {text-indent: -5ch;}
                   60% {text-indent: -4ch;}
                   70% {text-indent: -3ch;}
                   80% {text-indent: -2ch;}
                   90% {text-indent: -1ch;}
                   100% {text-indent: 0ch;} }
  img { width: 487px; height: 323px; border-radius: 5px; }
  body { font-size: 20px; font-weight: 500; }
  a { border-radius: 3px; padding: 5px; border: solid 1px black;
      background-color: lightskyblue; text-decoration: none; }
 </style></head>
<body><img src="img/B122.jpg"/>
<a href="http://order.jd.com/center/list.action">订单提交中
<dot>..........</dot></a></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，animation：dot 10s infinite step-start both 表示在 10 秒内完成 dot 动画，且无限重复，both 实际上是 animation-fill-mode：both。animation-fill-mode 属性规定动画在播放之前或之后的动画效果是否可见，该属性的语法格式如下。

```
animation-fill-mode: none | forwards | backwards | both
```

其中，各个属性值的意义如表 344-1 所示。

表 344-1　animation-fill-mode 属性值及意义

值	描　述
none	不改变默认行为
forwards	当动画完成后保持最后一个属性值(在最后一个关键帧中定义)
backwards	在 animation-delay 指定的一段时间内，在动画显示之前应用开始属性值(在第一个关键帧中定义)
both	向前和向后填充模式都被应用

10%｛text-indent：－9ch;｝表示动画完成 10%时文本缩进“－9ch”，即 10 个“.”仅显示一个“.”，其余以此类推。

此实例的源文件名是 myHtmlB123.html。

345　使用收缩、扩展的形式平滑切换两个场景

此实例主要通过设置元素的 transform 和 transition 属性实现以收缩、扩展的形式平滑切换两个场景。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针滑入左上角的图片，则图片将在指定的时间内扩展并滑入文本，如图 345-1 所示；如果鼠标指针滑入右下角的图片，则右下角的图片将在指定的时间内扩展并滑入文本，同时左上角的文本收缩并将在指定的时间内滑入图片；使用鼠标指

针在其他图片上滑入、滑出将实现类似的效果。有关此实例的主要代码如下。

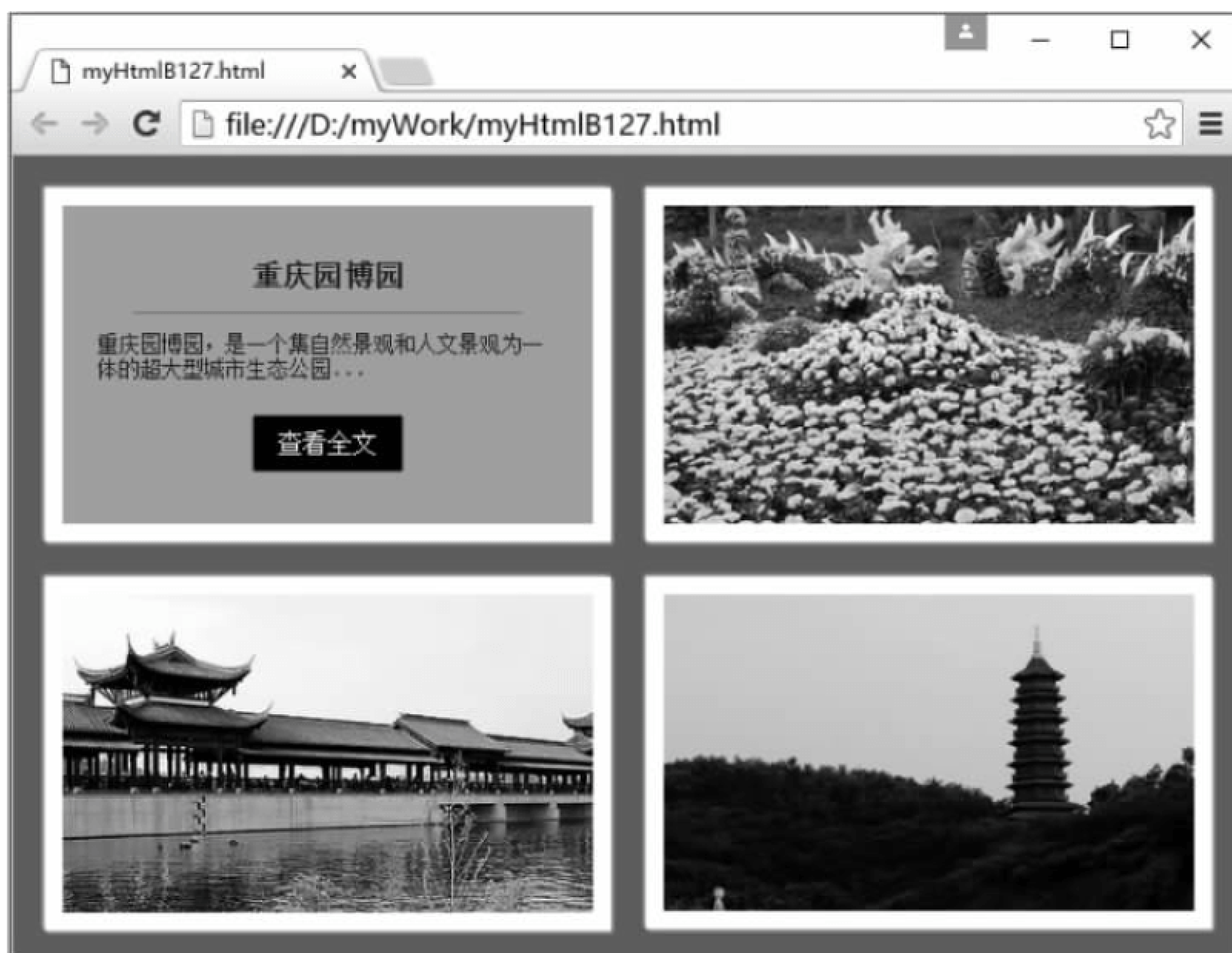


图 345-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .main *{ padding:0; margin:0;}
  .main { position: relative; width: 680px; margin: 0 auto;}
  .view { width: 300px; margin: 10px;float: left;
         border: 10px solid #FFF; overflow: hidden;
         position: relative;text-align: center;
         -webkit-box-shadow: 1px 1px 2px #E6E6E6, -1px -1px 2px #E6E6e6;
         cursor: default; background-color: aqua; }
  .view .mask{ width: 300px; height: 200px; position: absolute;
              overflow: hidden; top: 0; left: 0;}
  .view img {display: block; position: relative;max-width:100%;}
  .view h2 {text-transform: uppercase;color: #FFF;text-align: center;
           position: relative;font-size: 17px; padding: 10px;
           background: rgba(0, 0, 0, 0.8);margin: 20px 0 0 0;}
  .view p { font-size: 12px; position: relative; color: #FFF;
           padding: 10px 20px 20px; text-align: left;}
  .view .link { display: inline-block;text-decoration: none;
               padding: 7px 14px;background: #000;color: #FFF;
               text-transform: uppercase;
               -webkit-box-shadow: 0 0 1px #000;font-size: 14px;}
  .view .link:hover { -webkit-box-shadow: 0 0 5px #000;}
  .view-tenth img { -webkit-transform: scaleY(1);
                   -webkit-transition: all 0.7s ease-in-out;}
  .view-tenth .mask {background-color: rgba(249, 179, 255, 0.3);
                    -webkit-transition: all 0.5s linear; opacity: 0;}
```

```
.view-tenth h2 { border-bottom: 1px solid rgba(0, 0, 0, 0.3);
                 background: transparent; margin: 20px 40px 0px 40px;
                 -webkit-transform: scale(0); color: #333;
                 -webkit-transition: all 0.5s linear; opacity: 0;}
.view-tenth p {color: #333; opacity: 0; -webkit-transform: scale(0);
               -webkit-transition: all 0.5s linear;}
.view-tenth .link {opacity: 0; -webkit-transform: scale(0);
                   -webkit-transition: all 0.5s linear;}
.view-tenth:hover img { -webkit-transform: scale(10); opacity: 0; }
.view-tenth:hover .mask {opacity: 1;}
.view-tenth:hover h2,.view-tenth:hover p,.view-tenth:hover .link {
            -webkit-transform: scale(1); opacity: 1; }
body{ background-color: gray;}
</style></head>
<body><div class="main">
<div class="view view-tenth"><img src="img/B127A.jpg" />
  <div class="mask"><h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型
城市生态公园...</p><a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全
文</a></div></div>
 <div class="view view-tenth"><img src="img/B127B.jpg" /><div class="mask">
   <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...</p>
<a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div></div>
 <div class="view view-tenth"><img src="img/B127C.jpg" /><div class="mask">
  <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...</p>
<a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div></div>
 <div class="view view-tenth"><img src="img/B127D.jpg" /><div class="mask">
  <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...</p>
<a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，transform：scale(10)表示将元素(图片)放大10倍，transform：scale(0)则表示缩小至消失；transition：all 0.5s linear 表示以 linear 方式在0.5秒内完成所有属性的动画过渡。transition 属性是一个简写属性，用于设置 transition-property、transition-duration、transition-timing-function、transition-delay 几个过渡属性，默认形式为 transition：all 0 ease 0。

此实例的源文件名是 myHtmlB127.html。

346 以类似翻书的风格旋转切换两个场景

此实例主要通过使用 rotateY()等方法实现以类似翻书的风格旋转切换两个场景。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针滑入右上角的图片，则将在指定的时间内绕 Y 轴反向旋转90°滑入文本，如图346-1所示；如果鼠标指针滑入左下角的图片，则将在指定的时间内绕 Y 轴反向旋转90°并滑入文本，同时右上角的文本绕 Y 轴正向旋转90°并收缩文本以显示图片；使用鼠标指针在其他图片上滑入、滑出将实现类似的效果。



图 346-1

有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .main *{padding: 0; margin: 0; -webkit-box-sizing: border-box;}
  .main { position: relative; width: 640px; margin: 0 auto;}
  .view { width: 300px; margin: 10px; float: left; border: 10px solid #FFF;
          -webkit-box-shadow: 1px 1px 2px #E6E6E6, -1px -1px 2px #E6E6E6;
         cursor: default; }
  .view figure img {max-width: 100%; display: block; position: relative;}
  .view .mask { position: absolute; top: 0; left: 0;bottom: 0; padding: 10px;
               background-color: rgba(233, 194, 236, 0.9); color: black; }
  .view .mask h2 { margin: 0 0 5px; padding: 0 0 5px; color: black; font-size: 20px;
    font-weight: bold; text-align: center; border-bottom: 1px solid gray;}
  .view .mask p { font-size: 14px;}
  .view .link { position: absolute; bottom: 10px; right: 10px; text-align: center; padding: 5px 10px;
border-radius: 5px; display: inline-block; background: red; color: #FFF; text-decoration: none;}
  .view-tenth { -webkit-perspective: 1700px; -webkit-perspective-origin: 0 50%; }
  .view-tenth .mask { width: 100%; opacity: 0; -webkit-backface-visibility: hidden;
                      -webkit-transform-origin: 0 0; -webkit-transform: rotateY(-90deg);
                      -webkit-transition: -webkit-transform 0.4s, opacity 0.1s 0.3s; }
  .view-tenth figure:hover .mask {opacity: 1; -webkit-transform: rotateY(0deg);
                                  -webkit-transition: -webkit-transform 0.4s, opacity 0.1s; }
  body { background-color: gray; }
 </style></head>
<body><div class="main"><div class="view view-tenth">
 <figure><div><img src="img/B127A.jpg"/></div><div class="mask">
   <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...
</p><a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div>
</figure></div><div class="view view-tenth">
```

```
<figure><div><img src="img/B127B.jpg"/></div><div class="mask">
   <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...
</p><a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div>
</figure></div><div class="view view-tenth">
 <figure><div><img src="img/B127C.jpg"/></div><div class="mask">
  <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...
</p><a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div>
</figure></div><div class="view view-tenth">
 <figure><div><img src="img/B127D.jpg"></div><div class="mask">
 <h2>重庆园博园</h2><p>重庆园博园,是一个集自然景观和人文景观为一体的超大型城市生态公园...</p>
<a href="http://baike.so.com/doc/5639682-5852310.html" class="link">查看全文</a></div></figure>
</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,rotateY()表示从当前位置绕 Y 轴旋转到指定的角度。perspective 属性定义 3D 元素与视图的距离,以像素计,该属性允许改变 3D 元素以查看 3D 元素的视图,当为元素定义 perspective 属性时其子元素会获得透视效果,而不是元素本身,该属性通常与 perspective-origin 属性一起使用,这样就能够改变 3D 元素的底部位置。在实例中,如果把“-webkit-perspective:1700px;”改成“-webkit-perspective:100px;”,则会在测试时出现较强的动态效果。perspective 属性的语法格式如下。

```
perspective: number|none
```

其中,number 表示元素与视图的距离,以像素计;none 是默认值,与 0 相同,不设置透视。

perspective-origin 属性定义 3D 元素所基于的 X 轴和 Y 轴位置,该属性允许改变 3D 元素的底部位置。当为元素定义 perspective-origin 属性时其子元素会获得透视效果,而不是元素本身,该属性必须与 perspective 属性一起使用,而且只影响 3D 转换元素。perspective-origin 属性的语法格式如下。

```
perspective-origin: x-axis y-axis
```

其中,x-axis 定义该视图在 X 轴上的位置,默认值是 50%,可能的值有 left、center、right、length、%;y-axis 定义该视图在 Y 轴上的位置,默认值是 50%,可能的值有 top、center、bottom、length、%。

此实例的源文件名是 myHtmlB129.html。

347 创建渐变色文字的光影流动特效

此实例主要使用滚动的渐变色填充文字,从而实现光影流动的文字动画特效。当在 Google Chrome 浏览器中显示该页面时,文字的颜色将按照从左向右的方向渐变,由于渐变色是用动画不停变换的,因此像变色的探照灯不停地往复照射文字一样,如图 347-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .masked {background-image: -webkit-linear-gradient(left, #147B96, #E6D205 25%,
           #147B96 50%, #E6D205 75%, #147B96); -webkit-text-fill-color: transparent;
           -webkit-background-clip: text; -webkit-background-size: 200% 100%; -webkit
           -animation: masked-animation 4s infinite linear; font-size: 24px; margin: 60px;}
```

```
    @-webkit-keyframes masked-animation { 0% { background-position: 0 0;}
                                          100% { background-position: -100% 0; } }
  </style></head>
<body><div class="masked"><h1>待月西厢下,迎风户半开。隔墙花影动,疑是玉人来。</h1></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-image: -webkit-linear-gradient(left, #147B96, #E6D205 25%, #147B96 50%, #E6D205 75%, #147B96)、-webkit-text-fill-color: transparent、-webkit-background-clip: text 这些代码在一起表示用渐变色填充文字线条;-webkit-animation: masked-animation 4s infinite linear 表示在 4 秒内以线性方式执行 masked-animation 动画,且无限循环执行下去。

此实例的源文件名是 myHtmlB143.html。



图 347-1

348 不间断水平滚动显示序列中的广告图片

此实例主要设置 overflow、white-space、display 等属性,并使用定时器不断改变显示位置,从而实现不间断水平滚动播放无序列表中的广告图片的特效。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在图片上,则滚动播放暂停;如果鼠标指针离开图片,则继续滚动播放图片,效果如图 348-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script language="javascript">
  function ScrollImgLeft() {
   var speed=20;                                    //设置滚动速度
   var myBegin=document.getElementById("myBegin");
   var myEnd=document.getElementById("myEnd");
   var myBox=document.getElementById("myBox");
   myEnd.innerHTML=myBegin.innerHTML;
```

```
  function Marquee() {
   if (myEnd.offsetWidth - myBox.scrollLeft <= 0)
     myBox.scrollLeft -= myBegin.offsetWidth;
   else
     myBox.scrollLeft++;
  }
  var MyMar = setInterval(Marquee, speed);
  myBox.onmouseover = function() {              //鼠标指针悬浮时停止滚动
   clearInterval(MyMar);
  }
  myBox.onmouseout = function() {               //鼠标指针离开后继续滚动
    MyMar = setInterval(Marquee, speed);
 } }
 </script>
 <style type = "text/css">
  .myBox { width: 410px;  height: 210px;
          margin: 20px auto;                    /* 第二个 auto 能够实现盒子水平居中 */
          overflow: hidden;                     /* 超出部分隐藏 */
          white-space: nowrap; padding: 5px;    /* 不换行 */
          background: #FFFFFF; box-shadow: 1px 1px 10px #AAA;}
  img { width: 200px; height: 200px;border: 1px black solid;
        margin: auto 5px; box-shadow: 1px 1px 10px #AAA; }
   #myBegin, #myEnd, ul, li {  display: inline; }       /* 在一行中显示所有图像 */
 </style></head>
<body><div id = "myBox" class = "myBox"><div id = "myBegin">
 <ul><li><img src = "img/B202A.jpg"/></li><li><img src = "img/B202B.jpg"/></li>
     <li><img src = "img/B202C.jpg"/></li><li><img src = "img/B202D.jpg"/></li></ul></div>
 <div id = "myEnd"></div></div>
 <script type = "text/javascript">ScrollImgLeft();</script></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：inline 用于在一行中显示所有图片及其他元素；overflow：hidden 表示无序列表的所有列表项（图片）在显示窗口无法显示时隐藏超出范围的部分；white-space：nowrap 表示在显示窗口中的内容不能换行。在 CSS 中，white-space 属性用于设置如何处理元素内的空白，该属性支持的属性值的意义如下。

（1）normal：默认值，表示空白会被浏览器忽略。

（2）pre：表示空白会被浏览器保留，其行为方式类似 HTML 中的<pre>标签。



图　348-1

(3) nowrap：表示文本不会换行，文本会在同一行继续，直到遇到< br >标签为止。

(4) pre-wrap：表示保留空白符序列，但是正常地进行换行。

(5) pre-line：表示合并空白符序列，但是保留换行符。

(6) inherit：表示规定应该从父元素继承 white-space 属性的值。

此实例中的 JS 代码主要用于定时器函数 setInterval()改变图片的显示位置，以产生不断滚动的效果。

此实例的源文件名是 myHtmlB202.html。

349 使用计时器实现电话在桌面上震动的特效

此实例主要通过使用计时器函数 setTimeout()和 clearTimeout()不停地改变元素的左上角坐标值，从而实现类似于电话机在桌面上震动的特殊动画效果。当在 Google Chrome 浏览器中显示该页面时，单击"开始震动效果"按钮，则电话机就开始在办公桌上不停地震动(像地震发生时那样)，如图 349-1 所示；单击"停止震动效果"按钮，则震动结束。有关此实例的主要代码如下。



图 349-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  img { position: absolute; top: 30px;   z - index: 999; }
  div { width: 450px; height: 350px;background - image: url(img/B230.jpg);
        background - size: 100 % 100 % ; }
  input{ width:170px;}
 </style></head>
<body><center>
 <p><input type = "button" value = "开始震动效果" onclick = "myImage.start()"/>
    <input type = "button" value = "停止震动效果" onclick = "myImage.stop()"/></p>
 <div><img src = "img/B066A.png" id = "myImg"></div>
 <script type = "text/javascript">
  var m = document.getElementById("myImg");
```

```
    function SKclass(myPhone, Rate, speed) {
     var myLeft = myPhone.offsetLeft;
     var myTop = myPhone.offsetTop;
     this.stop = null; this.myTime = null; var myShake = this;
     this.start = function() {
      if (parseInt(myPhone.style.left) == myLeft - 2) {
        myPhone.style.top = myTop + 2 + "px";
        setTimeout(function() {
           myPhone.style.left = myLeft + 2 + "px"
        }, Rate)
        } else {
         myPhone.style.top = myTop - 2 + "px";
         setTimeout(function() {
           myPhone.style.left = myLeft - 2 + "px"
        }, Rate) }
        this.myTime = setTimeout(function() {
          myShake.start()
        }, speed); }
        this.stop = function() {  clearTimeout(this.myTime); } }
    var myImage = new SKclass(m, 20, 70);
   </script></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，left 和 top 表示电话机图像的左上角坐标值，办公桌是另外一幅图像；setTimeout()函数用于在指定的间隔时间内改变此坐标值。setTimeout()函数的语法格式如下。

```
setTimeout(code,millisec)
```

其中，参数 code 表示要执行的 JavaScript 代码串(此实例就是修改电话机图像的左上角坐标值)；参数 millisec 规定在执行代码前需要等待的毫秒数。

clearTimeout()函数则用于取消指定的 setTimeout()函数将要执行的代码，该函数的语法格式如下。

```
clearTimeout(id)
```

其中，参数 id 是 setTimeout()函数返回的 ID。

此实例的源文件名是 myHtmlB230.html。

350 使虚线边框线实现跑马灯滚动效果

此实例主要使用 setInterval()函数不断改变虚线风格的边框线的左上角坐标，从而使边框线实现跑马灯的滚动效果。当在 Google Chrome 浏览器中显示该页面时，图像周围的 4 条边框线的跑马灯滚动效果如图 350-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script><script type = "text/javascript">
   $(function() {
    setInterval(function() {     //动态改变 4 条边框线的左上角位置
      var $left = $(".dashed-top").css("left");
      var $top = $(".dashed-bottom").css("left");
       $left = parseInt($left); $top = parseInt($top);
```

```
        if ( $ left < 0) {
          $ left += 2;
        } else {
          $ left = - 1400;
        }
        if ( $ top > - 1000) {
          $ top -= 2;
        } else {
          $ top = 0;
        }
        $ (".dashed - top").css("left", $ left + "px");
        $ (".dashed - right").css("top", $ left + "px");
        $ (".dashed - bottom").css("left", $ top + "px");
        $ (".dashed - left").css("top", $ top + "px");
    }, 60); });
  </script>
  <style type = "text/css">
   .dashed - box { width: 400px; height: 280px; overflow: hidden; position: relative; }
   /* 上边框线 */
   .dashed - top { width: 2000px; height: 0px; border - bottom: 2px black dashed;
                   position: absolute; top: 0; left: - 1400px;}
   /* 左边框线 */
   .dashed - left { width: 0px; height: 2000px;border - left: 2px black dashed;
                    position: absolute; left: 0; top: - 1600px; }
   /* 下边框线 */
   .dashed - bottom { width: 2000px; height: 0px; border - bottom: 2px black dashed;
                      position: absolute; left: 0px; bottom: 0;}
   /* 右边框线 */
   .dashed - right { width: 0px; height: 2000px; border - left: 2px black dashed;
                     position: absolute; right: 0; top: - 1600px; }
  </style></head>
 <body><center><div class = "dashed - box"><img src = "img/B235.jpg">
   <div class = "dashed - top"></div><div class = "dashed - left"></div>
   <div class = "dashed - right"></div><div class = "dashed - bottom"></div></div>
 </center></body></html>
```



图 350-1

上面有底纹的代码是此实例的核心代码。在该部分代码中，border-bottom：2px black dashed 用于设置该 div 元素的底部边框线为 2px 的黑色虚线，由于该 div 元素的 height 为 0px，因此该 div 元素（上边框线）显示的即是一条虚线，而不是一个虚线围绕的方框，其他 3 条虚线的原理与之类似；top：0；left：－1400px 表示 div 元素（上边框线）的左上角坐标；setInterval()函数则主要用于在 JavaScript 代码中动态改变 4 条边框线的左上角坐标。setInterval()函数可按照指定的周期（以毫秒计）来调用函数或计算表达式。setInterval()函数会不停地调用代码，直到 clearInterval()函数被调用或窗口被关闭。由 setInterval()函数返回的 ID 值可用作 clearInterval()函数的参数。setInterval()函数的语法如下。

```
setInterval(code,millisec[,"lang"])
```

其中，参数 code 表示要调用的函数或要执行的代码串；参数 millisec 表示周期性执行或调用 code 之间的时间间隔，以毫秒计。该函数的返回值可传递给 clearInterval()函数，从而取消对 code 的周期性执行。

此实例的源文件名是 myHtmlB235.html。

351　使用定时器高仿改变进程的图文进度条

此实例主要在定时器 setInterval()函数中改变元素(div)的属性(width)，从而高仿改变进程的图文进度条的动态执行效果。当在 Google Chrome 浏览器中显示该页面时，绿色的进度条将从 0 逐渐填充到宽度值 300px，效果如图 351-1 所示。有关此实例的主要代码如下。

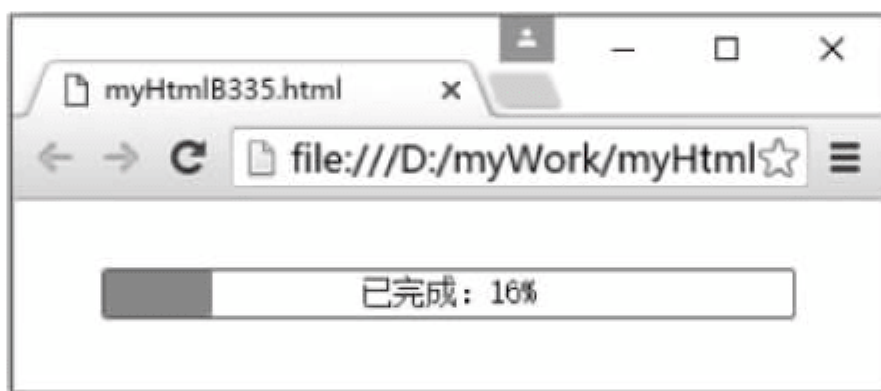


图　351-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
    $(document).ready(function() {
     var myWidth=0;
     var timer=setInterval(function() {//通过定时器模拟进度的改变
      if (myWidth==301) { return; }
      $(".myBackground").css("width", myWidth+"px");
      $("p").text("已完成: "+((myWidth/301)*100).toFixed(0)+"%");
       myWidth++;
   }, 10); });
 </script>
 <style type="text/css">
   /*设置进度条盒子样式*/
   .myBorder { margin: 30px auto;  width: 300px; height: 20px; border-radius: 2px;
               border: 1px solid green; text-align: center; overflow: hidden; }
   /*设置进度条填充部分样式*/
   .myBackground { width: 0px; height: 20px;
                   border-radius: 2px; background-color: #00FF00; }
```

```
    /* 设置进度指示文字样式 */
    p { margin: -19px;  font-size: 14px; }
  </style></head>
 <body><div class="myBorder"><div class="myBackground"></div>
     <p>已完成: 0%</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(".myBackground").css("width",myWidth+"px")用于设置已经完成部分的 div 块的宽度；setInterval()函数用于间隔指定的毫秒数不停地执行指定的代码，此处即是改变 div 块的宽度。

此实例的源文件名是 myHtmlB335.html。

352　通过 2D 旋转方式模拟走动的时钟表盘

此实例主要使用 setInterval()函数和 rotate()方法不断改变时、分、秒指针的旋转角度，从而实现动态走动的时钟表盘。当在 Google Chrome 浏览器中显示该页面时，动态走动的时钟表盘如图 352-1 所示。有关此实例的主要代码如下。

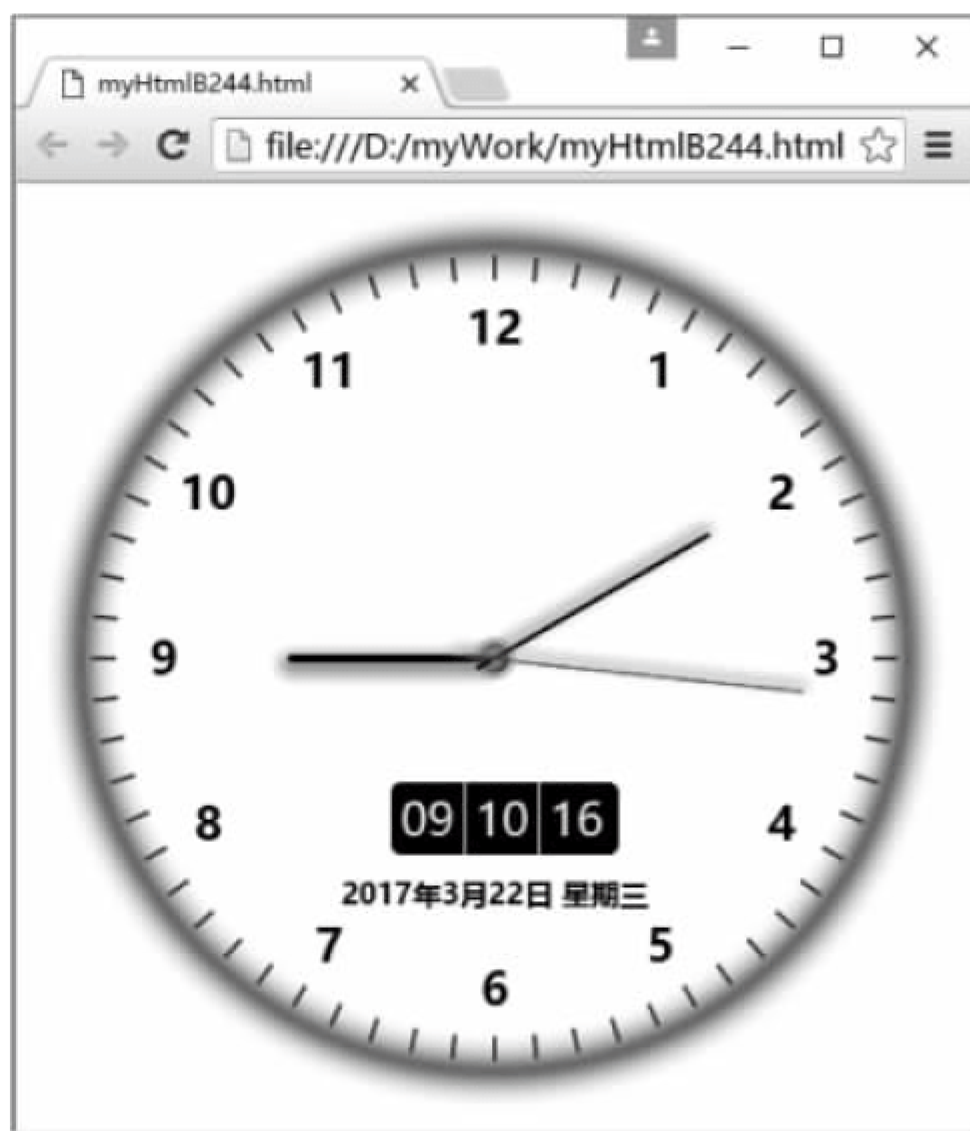


图　352-1

```
 <!doctype html><html><head><meta charset="UTF-8">
  <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
    $(function() {
     var clock = document.getElementById("clock");
     function initNumXY() {                        //1～12 数字的位置设置
      var radius = 160;
      var dot_num = 360/$(".dot").length;          //每个 div 对应的弧度数
      var ahd = dot_num * Math.PI/180;             //每个 dot 对应的弧度
      $(".dot").each(function(index, el) {
       $(this).css({"left": 180 + Math.sin((ahd * index)) * radius,
                    "top": 180 + Math.cos((ahd * index)) * radius });   });
```

```
    for (var i = 0; i < 60; i++) {                    //刻钟设置(一小时四刻钟)
     clock.innerHTML += "<div class = 'clock - scale'> " +
    "<div class = 'scale - hidden'></div>" + "<div class = 'scale - show'></div>" + "</div>"; }
    var scale = document.getElementsByClassName("clock - scale");
    for (var i = 0; i < scale.length; i++) {
     scale[i].style.transform = "rotate(" + (i * 6 - 90) + "deg)";  }  }
    initNumXY();                                      //显示小时刻度值
    //动态获取日期、时间信息
    var hour_line = document.getElementById("hour_line"),
    minute_line = document.getElementById("minute_line"),
    second_line = document.getElementById("second_line"),
    date_info = document.getElementById("date_info"),
    hour_time = document.getElementById("hour_time"),
    minute_time = document.getElementById("minute_time"),
    second_time = document.getElementById("second_time");
    function setTime() {                              //根据时间信息旋转时、分、秒针
     var nowdate = new Date();
     var year = nowdate.getFullYear(),                //获取年、月、日、时、分、秒
     month = nowdate.getMonth() + 1,
     day = nowdate.getDay(),
     hours = nowdate.getHours(),
     minutes = nowdate.getMinutes(),
     seconds = nowdate.getSeconds(),
     date = nowdate.getDate();
     var weekday = ["星期日", "星期一", "星期二", "星期三", "星期四", "星期五", "星期六"];
     date_info.innerHTML = year + "年" + month + "月" + date + "日   " + weekday[day];
     hour_time.innerHTML = hours >= 10 ? hours : "0" + hours;
     minute_time.innerHTML = minutes >= 10 ? minutes : "0" + minutes;
     second_time.innerHTML = seconds >= 10 ? seconds : "0" + seconds;
     //旋转时、分、秒针
     var hour_rotate = (hours * 30 - 90) + (Math.floor(minutes/12) * 6);
     hour_line.style.transform = 'rotate(' + hour_rotate + 'deg)';
     minute_line.style.transform = 'rotate(' + (minutes * 6 - 90) + 'deg)';
     second_line.style.transform = 'rotate(' + (seconds * 6 - 90) + 'deg)';
   }
   setInterval(setTime, 1000); });
  </script>
  <style type = "text/css">
    body, div, p { font - family: 'Microsoft Yahei'; font - size: 14px;}
    .myBox { width: 400px; height: 400px; border: 0px solid black;
            margin - top: 30px; border - radius: 50 %;position: relative;
            /* 设置刻度盘的外阴影 */
            box - shadow: 0px 0px 20px 3px #444; }
    .box { width: 400px;height: 400px; position: relative;
          border: 0px solid black; border - radius: 50 %;
          /* 设置刻度盘的内阴影 */
          box - shadow: 0px 0px 20px 3px #444 inset;}
    /* 中心点 */
    . origin { width: 20px; height: 20px; border - radius: 50 %; top: 190px; left: 190px; position:
absolute; background: - webkit - repeating - radial - gradient(yellow, green 5px, red 7px, white 8px); }
    /* 指针 */
    .clock_line { position: absolute;  z - index: 20;}
    /* 时针线 */
```

```
    .hour_line { width: 100px; height: 4px; top: 198px; left: 200px; background-color: black; border-
                radius: 2px; transform-origin: 0 50%;box-shadow: 1px -3px 8px 3px #AAA; }
    /*分针线*/
    .minute_line {width: 130px; height: 2px; top: 199px; left: 190px;
                 background-color: black;transform-origin: 7.692% 50%;
                 box-shadow: 1px -3px 8px 1px #AAA;}
    /*秒针线*/
    .second_line { width: 170px; height: 1px; top: 199.5px; left: 180px; background-color: red;
                   transform-origin: 11.765% 50%;box-shadow: 1px -3px 7px 1px #BBB; }
    .dot_box { width: inherit;  height: inherit; }
    /*时钟数*/
    .dot { width: 40px; height: 40px; line-height: 40px; text-align: center; font-size: 22px;
           position: absolute; font-weight: bold; }
    .clock-scale { width: 195px; height: 2px; transform-origin: 0% 50%;
                   z-index: 7;position: absolute; top: 199px; left: 200px;}
    .scale-show { width: 12px; height: 2px; background-color: #555; float: left;}
    .scale-hidden { width: 183px; height: 2px; float: left; }
    /*日期:2017年3月22日 星期三*/
    .date_info { width: 160px; height: 28px;line-height: 28px; text-align: center; position: absolute;
 z-index: 11; top: 300px; left: 120px;color: black; font-weight: bold; }
    /*以数字显示时、分、秒:大框 09 39 12*/
    .time_info { width: 110px;height: 35px;line-height: 35px; text-align: center; position: absolute;
 top: 260px;left: 150px;z-index: 11; color: white; background: black; border-radius: 5px;}
    /*以数字显示时、分、秒:小框 09 */
    .time { width: 35px; height: 35px;float: left; color: yellow;font-size: 22px;}
    #minute_time { border-left: 1px solid white;border-right: 1px solid white; }
   </style></head>
 <body><center><div class="myBox"><div class="box" id="clock">
    <div class="origin"></div><div class="dot_box">
     <div class="dot">6</div><div class="dot">5</div><div class="dot">4</div>
     <div class="dot">3</div><div class="dot">2</div><div class="dot">1</div>
     <div class="dot">12</div><div class="dot">11</div><div class="dot">10</div>
     <div class="dot">9</div><div class="dot">8</div>
     <div class="dot">7</div></div>
   <!-- 时、分、秒针 -->
   <div class="clock_line hour_line" id="hour_line"></div>
   <div class="clock_line minute_line" id="minute_line"></div>
   <div class="clock_line second_line" id="second_line"></div>
   <!-- 日期 -->
   <div class="date_info" id="date_info"></div>
   <!-- 时间 -->
   <div class="time_info"><div class="time" id="hour_time"></div>
    <div class="time" id="minute_time"></div>
    <div class="time" id="second_time"></div></div></div></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,getFullYear()方法用于获取一个表示年份的4位数字;getMonth()方法用于获取表示月份的数字;getDate()方法用于获取月份的某一天的数字;getDay()方法用于获取表示星期的某一天的数字;getHours()方法用于获取时间的小时字段;getMinutes()方法用于获取时间的分钟字段;getSeconds()方法用于获取时间的秒数;setInterval()函数可按照指定的周期(以毫秒计)来调用函数或计算表达式。上述内容属于JavaScript部分。在CSS3中,transform属性负责向元素应用2D或3D转换,该属性允许对元素进行旋转、缩放、移动或倾斜。在此实例中,hour_line. style. transform='rotate('+hour_rotate+'deg)'即是通过

指定的小时数(通过 JavaScript 获取)旋转时针;setInterval(setTime,1000)则用于每隔 1 秒执行旋转和检测时、分、秒的动作。

此实例的源文件名是 myHtmlB244.html。

353 在单行菜单中以下拉方式滑出焦点菜单

此实例主要使用 jQuery 的 animate()方法改变元素的 top 值,从而实现在单行菜单中以下拉方式滑出焦点菜单。当在 Google Chrome 浏览器中显示该页面时,如果鼠标指针悬浮在"行政审批"菜单项上,则将从上向下滑出蓝色的焦点菜单"行政审批",如图 353-1 所示。当然,如果鼠标指针悬浮在其他菜单项上,也能实现类似的效果。有关此实例的主要代码如下。



图 353-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(document).ready(function() {
   $("#myMenu li a").wrapInner('<span class="out"></span>');
   $("#myMenu li a").each(function() {
    $('<span class="over">'+$(this).text()+'</span>').appendTo(this);  });
  //响应鼠标指针悬浮于每个菜单项
   $("#myMenu li a").hover(function() {         //鼠标指标进入时响应
    $(".out", this).stop().animate({'top': '48px'}, 300);
    $(".over", this).stop().animate({'top': '0px'}, 300);
  }, function() {                                //鼠标指标离开时响应
    $(".out", this).stop().animate({'top': '0px'}, 300);
    $(".over", this).stop().animate({'top': '-48px'}, 300); });});
 </script>
 <style type="text/css">
  .menu { height: 48px; display: block; padding: 0px;
         width: auto; margin: 5px auto 0 auto; }
  .menu ul { list-style: none; padding: 0; margin: 0; }
  .menu ul li { float: left; overflow: hidden; position: relative;
              line-height: 48px; text-align: center; margin-right: 1px; }
  .menu ul li a { position: relative;  display: block; width: 111px; height: 48px; font-size: 18px; text
                -decoration: none; cursor: pointer; line-height: 48px; font-weight: bold; }
  .menu ul li a span { position: absolute; left: 0; width: 111px;}
  .menu ul li a span.out { top: 0px; }
  .menu ul li a span.over, .menu ul li a span.bg { top: -48px;}
  #myMenu { background-color: red; overflow: hidden; }
  #myMenu ul li a { color: yellow;  }
  #myMenu ul li a span.over { background-color: blue; }
 </style></head>
<body><div id="myMenu" class="menu">
```

```
<ul><li><a href="#">政府信息</a></li><li><a href="#">街镇部门</a></li>
    <li><a href="#">行政审批</a></li><li><a href="#">互动交流</a></li></ul></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,animate({'top':'48px'},300)表示在300毫秒内将菜单的top属性值由之前的数字改变为48px,即菜单向下滑出48px。在jQuery中,animate()方法执行CSS属性集的自定义动画,该方法有两种形式,实例所使用形式的语法格式如下。

```
$(selector).animate(styles,options)
```

其中,参数styles规定产生动画效果的CSS样式和值;参数options是可选参数,规定动画的额外选项。

可能的值如下。

(1) speed:设置动画的速度,可能的值为毫秒(比如300)、slow、normal、fast,例如animate({'top':'48px'},"slow")。

(2) easing:规定要使用的easing函数。

(3) callback:规定动画完成之后要执行的函数。

(4) step:规定动画的每一步完成之后要执行的函数。

(5) queue:布尔值,指示是否在效果队列中放置动画,如果为false,则动画将立即开始。

(6) specialEasing:来自styles参数的一个或多个CSS属性的映射,以及它们的对应easing函数。

stop()方法用于停止当前正在运行的动画;appendTo()方法用于在被选元素的结尾(仍然在内部)插入指定内容。

此实例的源文件名是myHtmlB248.html。

354 在延迟指定时间后执行显示或隐藏元素

此实例主要通过设置visibility和transition-delay属性实现在延迟指定时间后执行显示或隐藏元素。当在Google Chrome浏览器中显示该页面时,根据超链接"鼠标在此停留5秒,即出现美景;鼠标离开此地3秒,美景即消失"的提示执行鼠标操作,即可实现文字所述的功能,如图354-1所示。有关此实例的主要代码如下。



图 354-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .trans-delay { -webkit-transition: visibility 2s linear; }
  .image-delay { margin: 5px; visibility: hidden; -webkit-transition-delay: 1s; }
  .hover-delay:hover .image-delay { visibility: visible;
                                    -webkit-transition-delay: 3s; }
  img{ border-radius: 10px;}
 </style></head>
<body><center><div class="hover-delay">
  <a href="#">鼠标在此停留 5 秒,即出现美景;鼠标离开此地 3 秒,美景即消失</a>
  <img src="img/B156A.jpg" class="trans-delay image-delay"/></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,transition-delay 属性规定过渡效果何时开始,其值以秒或毫秒计;-webkit-transition-delay:3s 表示延迟 3 秒过渡到下一状态。visibility 属性用于设置或检索是否显示对象(元素),与 display 属性不同,此属性为隐藏的对象保留其占据的物理空间。

visibility 属性的语法格式如下。

```
visibility:visible | hidden | collapse
```

其中,visible 属性值用于设置对象可视;hidden 属性值用于设置对象隐藏;collapse 属性值主要用来隐藏表格的行或列,隐藏的行或列能够被其他内容使用,对于表格外的其他对象,其作用等同于 hidden。

此实例的源文件名是 myHtmlB175.html。

355 使用 gif 格式的图像实现边框线跑马灯效果

此实例主要通过使用 gif 动画图像实现边框线的跑马灯走动效果。当在 Google Chrome 浏览器中显示该页面时,黑白相间的边框线像跑马灯一样不停走动,如图 355-1 所示。有关此实例的主要代码如下。

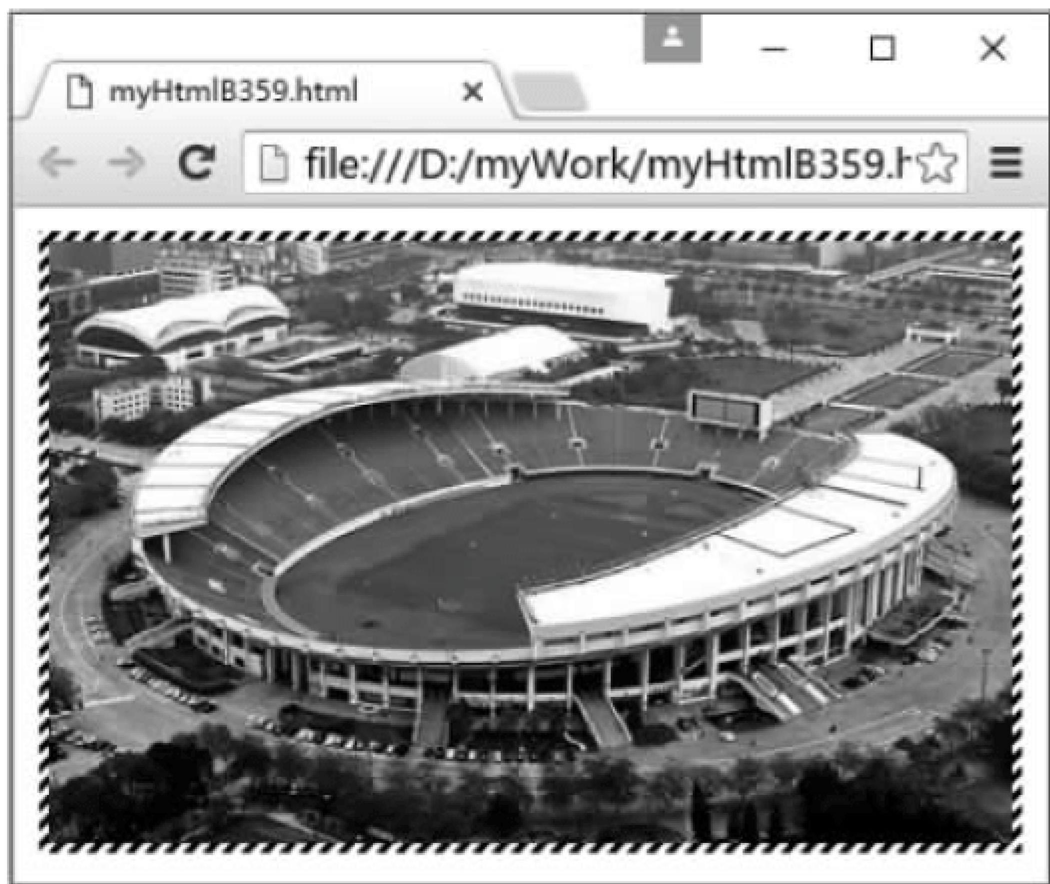


图 355-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { margin: 10px auto; overflow: hidden;
          /* 盒子的宽度和高度始终比图像大 */
          width: 408px; height: 258px; background: url(img/B359.gif); }
  .myImage { position: absolute; margin: 4px 4px; display: block;
           /* 图像的宽度和高度 */
           width: 400px; height: 250px;background: url(img/B359.jpg) no-repeat;}
 </style></head>
<body><div class="myBox"> <div class="myImage"> </div> </div></body></html>
```

上面有底纹的代码是此实例的核心代码。此实例的代码比较简单，就是在如图 355-2 所示的 gif 动画图像上面叠加一幅普通图像，因为动画图像比普通图像稍大，多出的部分就像是普通图像的边框线，边框线的跑马灯走动效果则是由 gif 动画图像自身的变化实现的。

此实例的源文件名是 myHtmlB359.html。

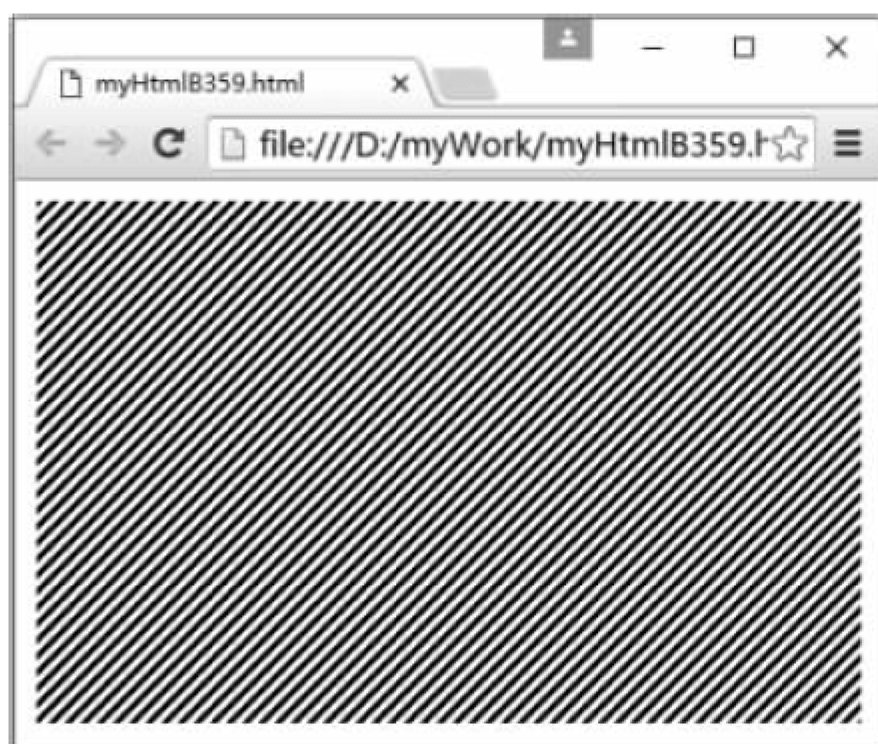


图 355-2

356 在页面中嵌入播放 swf 等格式的动画文件

此实例主要通过使用<embed>标签实现在页面中嵌入播放 swf 等格式的动画文件。当在浏览器中显示该页面时将自动播放动画文件 A003.swf，如图 356-1 所示。有关此实例的主要代码如下。



图 356-1

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><section align="center"> <embed src="media/A003.swf" width="400px" height="100px"/>
</section> </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<embed>标签用于定义嵌入的内容，比如插件，该标签是 HTML5 中的新标签。<embed>标签的属性说明如表 356-1 所示。

表 356-1 <embed>标签的属性及说明

属 性	值	描 述
height	pixels	设置嵌入内容的高度
src	url	嵌入内容的 URL
type	type	定义嵌入内容的类型
width	pixels	设置嵌入内容的宽度

此实例的源文件名是 myHtmlA003.html。

357 使用 SVG 矢量图形模拟水波特效文字

此实例主要通过使用 SVG 矢量图形实现水波动画的特效文字。当在 Google Chrome 浏览器中显示该页面时，青色的水波不停地在文字上晃动，如图 357-1 所示。有关此实例的主要代码如下。



图 357-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*设置页面背景的基本样式*/
  body { text-align: center; background: lightgrey;
        background-size: cover; background-position: 50%; }
 </style></head>
<body><div align="center">
 <svg x="0px" y="0px" width="720px" height="250px" xml:space="preserve">
  <defs><pattern id="water" width="0.25" height="1.1" patternContentUnits=
"objectBoundingBox">
    <path fill="#000" d="M0.25,1H0c0,0,0-0.659,0-0.916c0.083-0.303,0.158,0.334,
0.25,0C0.25,0.327,0.25,1,0.25,1z"/></pattern>
    <text id="text" transform="translate(-2,200)" font-family="微软雅黑"
font-size="120">炫酷实例集锦</text>
    <mask id="text-mask"><use x="0" y="0" xlink:href="#text" opacity="1" fill="cyan"/></mask>
    <g id="eff"><use x="0" y="0" xlink:href="#text" fill="cyan"/>
       <rect class="water-fill" mask="url(#text-mask)" fill="url(#water)" x="-300"
```

```
y = "150" width = "1200" height = "180" opacity = "0.5">
         < animate attributeType = " xml" attributeName = " x" from = " - 300" to = " 0" repeatCount =
" indefinite" dur = "2s"/></rect>
      <rect class = "water - fill" mask = "url( # text - mask)" fill = "url( # water)" y = "145" width =
"1600" height = "180" opacity = "0.5">
         < animate attributeType = " xml" attributeName = " x" from = " - 400" to = " 0" repeatCount =
" indefinite" dur = "3s"/></rect>
      <rect class = "water - fill" mask = "url( # text - mask)" fill = "url( # water)" y = "155" width =
"800" height = "180" opacity = "0.5">
         < animate attributeType = " xml" attributeName = " x" from = " - 200" to = " 0" repeatCount =
" indefinite" dur = "1.4s"/></rect>
      <rect class = "water - fill" mask = "url( # text - mask)" fill = "url( # water)" y = "155" width =
"2000" height = "180" opacity = "0.5">
         < animate attributeType = " xml" attributeName = " x" from = " - 500" to = " 0" repeatCount =
" indefinite" dur = "2.8s"/></rect></g></defs>
      <use xlink:href = " # eff" opacity = "0.9" style = "mix - blend - mode:color - burn"/>
</svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，标签< svg ></svg >中的代码是符合SVG规范的代码。SVG是可伸缩矢量图形(Scalable Vector Graphics)的缩写，它使用XML格式定义图形图像，图像在放大或改变尺寸的情况下质量不会有所损失，SVG与诸如DOM和XSL之类的W3C标准是一个整体。Internet Explorer 9、火狐、Google Chrome、Opera和Safari浏览器都支持SVG。在Google Chrome浏览器中可以直接在HTML代码中插入SVG代码。

此实例的源文件名是myHtmlA181.html。

358 在SVG中实现文字沿三角形的边线跑动

此实例主要通过使用SVG的animateMotion动画实现文字沿着三角形的3条边线逆时针跑动。当在Google Chrome浏览器中显示该页面时，文字“酷”立即沿着三角形的3条边线逆时针跑动，图358-1所示的效果是“酷”字在右边线的情形。有关此实例的主要代码如下。

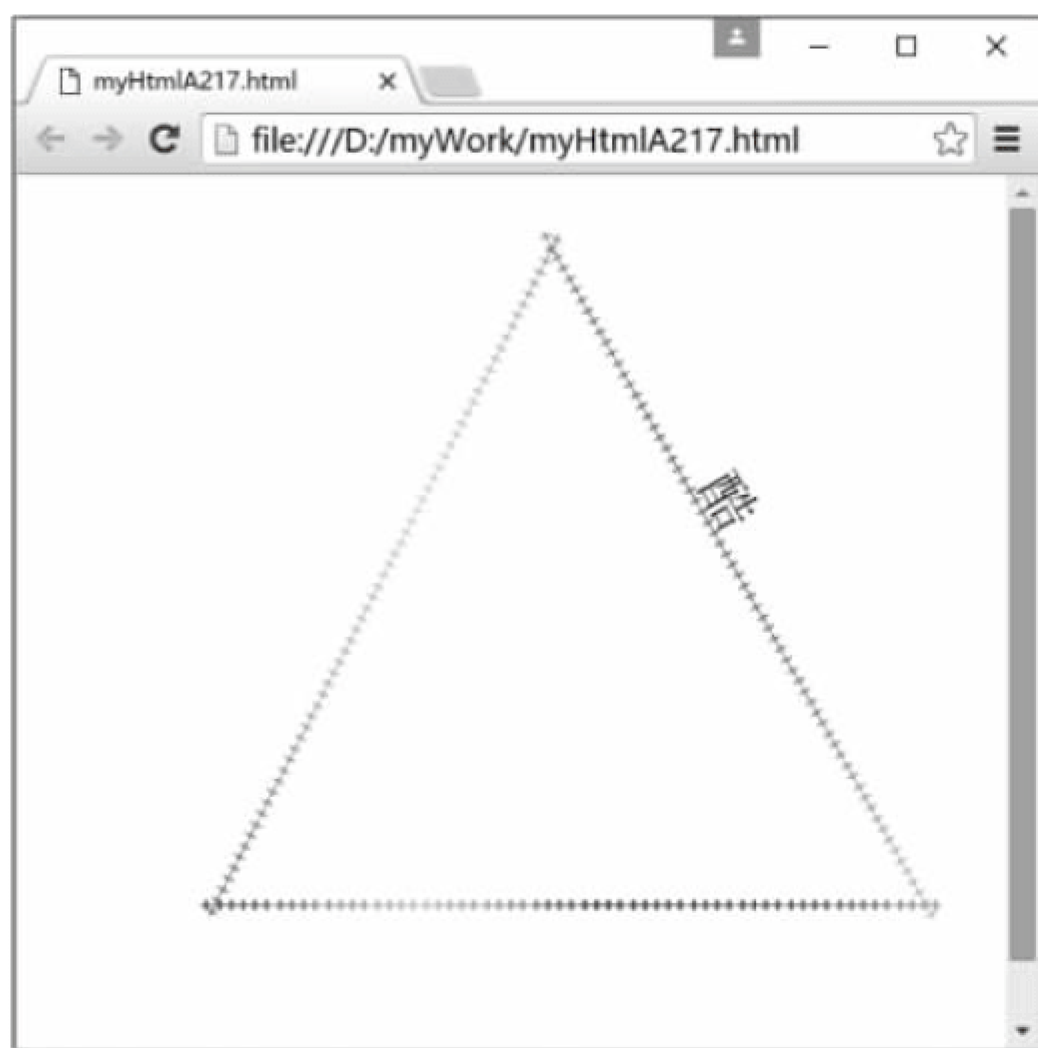


图 358-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  .svg-defs { position: absolute; }
  .texts { text-align: center; width:460px; height:460px; }
  /*设置文字的基本样式*/
  .myText { fill: url(#p-stripes); font-size: 5px; font-weight: bold; }
 </style></head>
<body><div align = "center">
 <svg class = "svg-defs"><g><defs>
  <!--设置路径是一个三角形-->
  <path id = "MyPath1" d = "M 250 20 L 70 370 L 450 370 Z" transform = "scale(1)"/>
  <!--设置文字符号在三角形路径上的渐变颜色-->
  <linearGradient id = "MyGradient" x1 = "0" y1 = "0" x2 = "100%" y2 = "0%">
   <stop offset = "0%" stop-color = "crimson"/>
   <stop offset = "10%" stop-color = "red"/>
   <stop offset = "20%" stop-color = "orangered"/>
   <stop offset = "30%" stop-color = "orange"/>
   <stop offset = "40%" stop-color = "gold"/>
   <stop offset = "50%" stop-color = "yellowgreen"/>
   <stop offset = "60%" stop-color = "green"/>
   <stop offset = "70%" stop-color = "steelblue"/>
   <stop offset = "80%" stop-color = "mediumpurple"/>
   <stop offset = "90%" stop-color = "purple"/></linearGradient>
   <pattern id = "p-stripes" patternUnits = "userSpaceOnUse" width = "100%" height = "100%">
   <rect width = "460" height = "460" fill = "url(#MyGradient)"/>
   </pattern></defs></g></svg>
  <svg class = "texts">
   <text class = "myText" transform = "translate(0)">
    <textPath xlink:href = "#MyPath1">++++++++++++++++++++++++++++++++++++++++++++++++ + ++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ + </textPath></text>
    <g transform = "translate(0,0)"><text id = "TextElement" x = "0" y = "0" style = "font-family:
Verdana;font-size: 29px">酷
      <animateMotion path = "M 250 20 L 70 370 L 450 370 Z" dur = "5s" fill = "freeze" rotate =
"auto-reverse"/></text>
   </g></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<animateMotion path="M 250 20 L 70 370 L 450 370 Z" dur="5s" fill="freeze" rotate="auto-reverse"/>用于实现文字的跑动。其中,path="M 250 20 L 70 370 L 450 370 Z"用于指明文字的跑动路径是一个三角形,此三角形本身是不可见的,为了便于在插图中显示这种效果,前面用专门的代码绘制了此三角形;rotate="auto-reverse"用于使文字“酷”在不同的位置自动调整自己的方向;dur="5s"表示动画的持续时间是5秒;fill="freeze"表示在动画结束后文字“酷”自动停留在最后的位置。

此实例的源文件名是myHtmlA217.html。

359 在SVG中实现文字沿螺旋线跑动消失

此实例主要通过使用SVG的animate动画实现文字沿着展开的螺旋线跑动消失。当在Google Chrome浏览器中显示该页面时,文字“关关雎鸠…”立即沿着螺旋线轨迹跑动(设置stroke="red"即可看到红色的螺旋线),如图359-1所示。有关此实例的主要代码如下。



图 359-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .mySVG { text-align: center;
          width: 460px; height: 460px; }
   #myTextPath {font-family: Verdana;font-size: 28px;}
 </style></head>
<body><div align="center">
 <svg class="mySVG">
  <path id="myPath" d="M153 334  C153 334 151 334 151 334
   C151 339 153 344 156 344  C164 344 171 339 171 334  C171 322 164 314 156 314
   C142 314 131 322 131 334  C131 350 142 364 156 364   C175 364 191 350 191 334
   C191 311 175 294 156 294  C131 294 111 311 111 334   C111 361 131 384 156 384
   C186 384 211 361 211 334  C211 300 186 274 156 274"
   stroke="white"  fill="none"  transform="scale(3),translate(-90,-270)"/>
  <text><textPath id="myTextPath" xlink:href="#myPath">
    关关雎鸠,在河之洲。窈窕淑女,君子好逑。参差荇菜,左右流之。窈窕淑女,寤寐求之。求之不得,寤寐思服。悠哉悠哉,辗转反侧。参差荇菜,左右采之。窈窕淑女,琴瑟友之。参差荇菜,左右芼之。窈窕淑女,钟鼓乐之。</textPath></text>
  <animate xlink:href="#myTextPath" attributeName="startOffset" from="0%" to="100%" begin="0s"  dur="25s" repeatCount="indefinite" keyTimes="0;1" calcMode="spline" keySplines="0.1 0.2 0.22 1"/></svg>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<path id="myPath" d="M153 334 C153 334 151 334 151 334 C151 339 153 344 156 344 C164 344 171 339 171 334 C171 322 164 314 156 314 C142 314 131 322 131 334 C131 350 142 364 156 364 C175 364 191 350 191 334 C191 311 175 294 156 294 C131 294 111 311 111 334 C111 361 131 384 156 384 C186 384 211 361 211 334 C211 300 186 274 156 274" stroke="white" fill="none" transform="scale(3),translate(-90,-270)"/>用于以路径的方式创建螺旋线,d 属性值创建的螺旋线其实非常小,因此使用了 transform="scale(3),translate(-90,-270)"进行放大平移处理;stroke="white"用于设置螺旋线的颜色,此处设置为与背景相同的白色,实际上就是不显示。

<animate xlink:href="#myTextPath" attributeName="startOffset" from="0%" to="100%" begin="0s" dur="25s" repeatCount="indefinite" keyTimes="0;1" calcMode="spline" keySplines="0.1 0.2 0.22 1"/>用于配置动画参数,from属性规定动画起始值,如果起始值和默认值相同,from可以省略;to属性规定动画的终止值;dur属性表示持续时间;calcMode属性表示动画在持续时间中的工作模式;attributeName属性规定元素的哪个属性会产生动画效果,此实例即是字符串的开始位置。

此实例的源文件名是myHtmlA218.html。

360　使用SVG的animateTransform旋转图像

此实例主要在SVG的<image>标签中嵌入<animateTransform>标签,从而实现<image>标签代表的图像根据<animateTransform>标签指定的属性值进行旋转的效果。当在Google Chrome浏览器中显示该页面时,单击图像,图像则自动围绕中心旋转一周,如图360-1所示。有关此实例的主要代码如下。

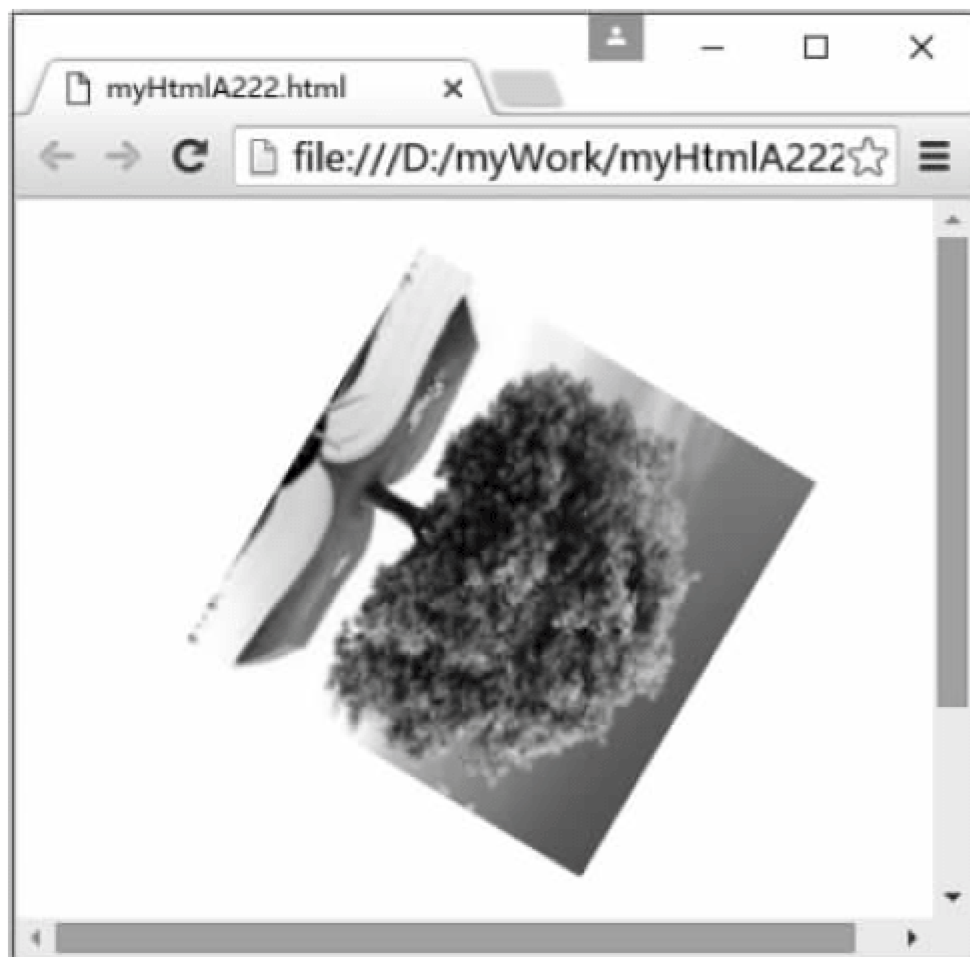


图　360-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("image").click(function() {    //单击图像即可旋转图像
    var myAnimate = document.getElementsByTagName("animateTransform")[0];
    myAnimate.beginElement();
  });});
 </script>
 <style type="text/css">
  .mySVG { width: 410px;height: 410px;}
 </style></head>
<body><div align="center"><svg class="mySVG">
  <image width="200" height="200" x="103" y="50"  xlink:href="img/A222.jpg">
  <animateTransform dur="2s" attributeName="transform" type="rotate" from="0,203,150" to="360,
203,150"/></image></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<animateTransform dur="2s" attributeName="transform" type="rotate" from="0,203,150" to="360,203, 150"/>表示在2秒的时间内以(203, 150)点为中心旋转360°。如果需要让旋转动画永不停歇,则应该设置repeatCount属性值为indefinite,即<animateTransform dur="2s" attributeName="transform" repeatCount="indefinite" type="rotate" from="0,203, 150" to="360,203,150"/>。

此实例的源文件名是myHtmlA222.html。

361 使用SVG的animate组合多种动画特效

此实例主要在SVG的<image>标签中嵌入多个<animate>标签,从而实现<image>标签代表的足球根据<animate>标签指定的属性值改变位置和透明度。当在Google Chrome浏览器中显示该页面时,足球将从左下角向右上角飞去,在飞去的过程中透明度将逐渐减小,直到消失,当足球接近右上角的时候效果如图361-1所示。有关此实例的主要代码如下。

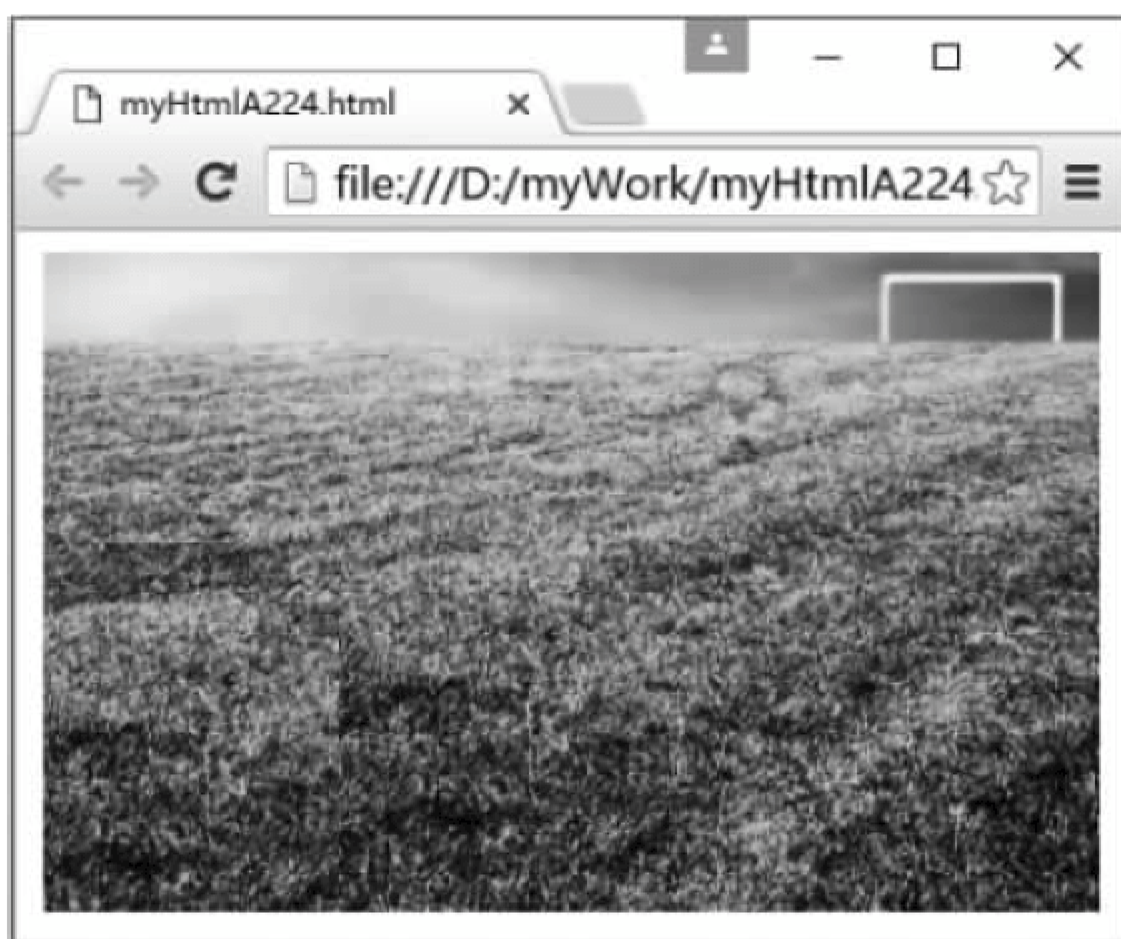


图 361-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
 .mySVG { width: 400px;height: 250px;
          background: url(img/B340.jpg);background-size: auto;}
 </style></head>
<body><div align="center">
 <svg class="mySVG">
  <image width="35" height="35" x="30" y="150" xlink:href="img/B255.png">
   <animate attributeName="x" from="30" to="330" begin="0s" dur="5s" repeatCount="indefinite"/>

   <animate attributeName="y" from="150" to="0" begin="0s" dur="5s" repeatCount="indefinite"/>
   <animate attributeName="opacity" from="1" to="0" begin="0s" dur="5s" repeatCount=
"indefinite"/></image></svg></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<animate attributeName="x" from="30" to="330" begin="0s" dur="5s" repeatCount="indefinite"/>表示在5秒的时间内在水平方向

上足球从 30px 移到 330px；<animate attributeName="y" from="150" to="0" begin="0s" dur="5s" repeatCount="indefinite"/>表示在 5 秒的时间内在垂直方向上足球从 150px 移到 0px；<animate attributeName="opacity" from="1" to="0" begin="0s" dur="5s" repeatCount="indefinite"/>表示在 5 秒的时间内足球的透明度从 1 渐变为 0。

此实例的源文件名是 myHtmlA224.html。

362 使用 SVG 的 animateMotion 模拟过山车

此实例主要在 SVG 的<image>标签中嵌入<animateMotion>标签，从而实现<image>标签代表的小车沿着<animateMotion>标签的属性值指定的过山车线路前进的效果。当在 Google Chrome 浏览器中显示该页面时，小车将沿着绿色的过山车线路前进，如图 362-1 所示。有关此实例的主要代码如下。

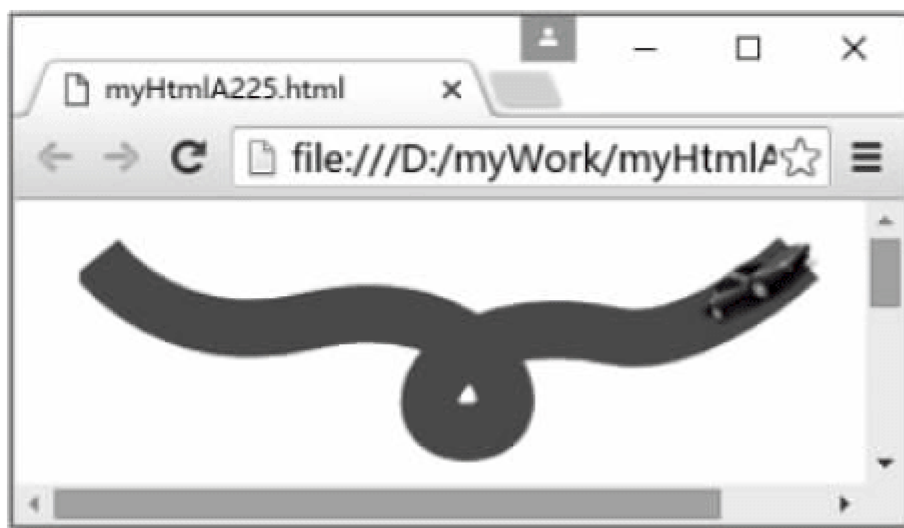


图 362-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .mySVG { width: 400px; height: 350px; }
  div{ margin-top: 10px; margin-left: 20px;}
 </style></head>
<body><div><svg class="mySVG"  >
  <path fill="none" stroke="green" stroke-width="25px" d="M7.4,15.3c17,20.4, 48.8,38,91.6,
27.8c79.5-18.9,107.4,48.2,69.4,48.2c-33.9,0-15.2-58.1,65.4-41.7c26.2,5.3,63.2-19.1,79.1-
34.3"  />
  <image width="55" height="55"  x="-35" y="-30" xlink:href="img/A225.png">
   <animateMotion dur="15s" path="M7.4,15.3c17,20.4,48.8,38,91.6,27.8c79.5-18.9, 107.4,48.2,
69.4,48.2c-33.9,0-15.2-58.1,65.4-41.7c26.2,5.3,63.2-19.1,79.1-34.3"     repeatCount=
"indefinite" rotate="auto" /></image></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<animateMotion dur="15s" path="M7.4,15.3c17,20.4,48.8,38,91.6,27.8c79.5-18.9,107.4,48.2,69.4,48.2c-33.9, 0-15.2-58.1, 65.4-41.7c26.2,5.3,63.2-19.1,79.1-34.3" repeatCount="indefinite" rotate="auto"/>中的 dur 用于定义动画播放的总时间；path 用于定义动画的工作路径，小车的位置以 path 的起点 M 开始计算，也就是会把 M 当作小车的(0,0)坐标；repeatCount 用于定义重复播放次数，indefinite 表示无限重播；rotate 表示小车在前进的过程中是否根据路径自动翻转。

此实例的源文件名是 myHtmlA225.html。

363　使用 SVG 的 animateMotion 模拟老鼠逃跑

此实例主要在 SVG 的< image >标签中嵌入< animateMotion >标签，从而实现< image >标签代表的老鼠沿着< animateMotion >标签的 path 属性值指定的线路逃跑的效果。当在 Google Chrome 浏览器中显示该页面时，老鼠将沿着封闭的曲线逃跑，如图 363-1 所示。有关此实例的主要代码如下。

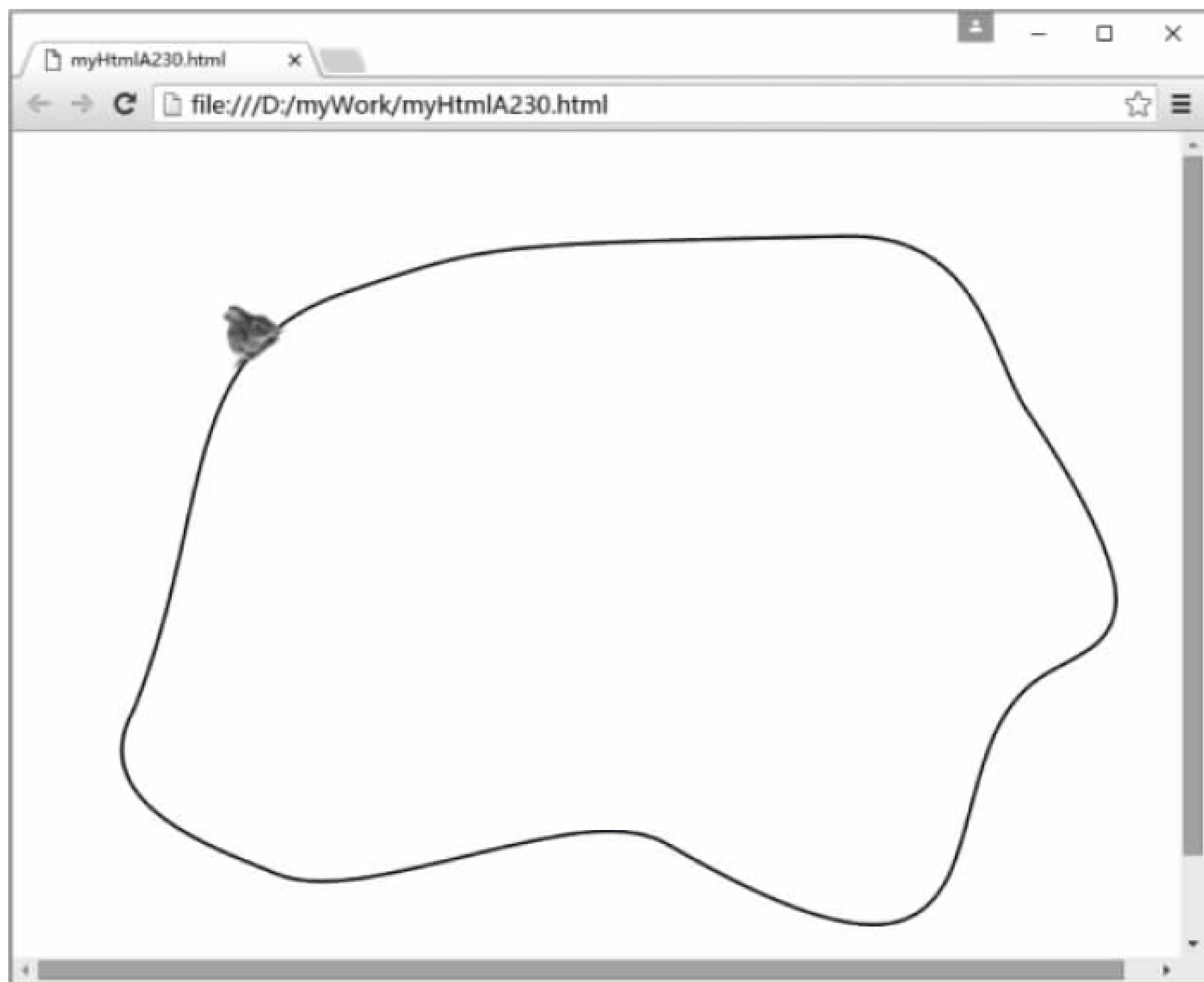


图　363-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .mySVG { width: 800px; height: 600px;}
  div { margin - top: 2px; margin - left: 2px;}
 </style></head>
<body><div><svg class = "mySVG">
  <path fill = "white" stroke = "black" stroke - width = "2px"
     d = "M215,100.3c97.8 - 32.6,90.5 - 31.9,336 - 37.6c92.4 - 2.1,98.1,81.6,121.8,116.4
c101.7,149.9,53.5,155.9,14.7,178c - 96.4,54.9,5.4,269 - 257,115.1c - 57 - 33.5 - 203,46.3 - 263.7,
20.1c - 33.5 - 14.5 - 132.5 - 45.5 - 95 - 111.1C125.9,246.6,98.6,139.1,215,100.3z"/>
  < image width = "45" height = "45" x = " - 35" y = " - 35" xlink:href = "img/A130.png">
   <animateMotion dur = "15s" path = "M215,100.3c97.8 - 32.6,90.5 - 31.9,336 - 37.6 c92.4  - 2.1,98.1,
81.6,121.8,116.4c101.7,149.9,53.5,155.9,14.7,178c - 96.4,54.9,5.4,269 - 257,115.1c - 57 - 33.5 - 203,
46.3 - 263.7,20.1c - 33.5 - 14.5 - 132.5 - 45.5 - 95 - 111.1C125.9,246.6,98.6,139.1,215,100.3z"
repeatCount = "indefinite" rotate = "auto"/>
 </image></svg></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< animateMotion >标签中的 dur 用于定义动画（老鼠逃跑）执行一次（一周）的时间；path 用于定义老鼠逃跑的线路；repeatCount 用于定义重复播放次数，indefinite 表示无限重播；rotate 表示老鼠在逃跑的过程中是否根据线路自动翻转（自身的姿势）。

此实例的源文件名是 myHtmlA230.html。

364　为超链接提示框实现爆炸式的关闭风格

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 explode，从而实现为超链接提示框添加爆炸式的关闭风格。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将显示工具提示框；当鼠标指针离开超链接 500 毫秒之后，则将以爆炸式风格关闭该工具提示框，效果如图 364-1 所示。有关此实例的主要代码如下。



图　364-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myExplode1").tooltip({//以爆炸的风格关闭$("#myExplode1")的提示框
              hide: { effect: "explode", delay:500 } });
   $("#myExplode2").tooltip({ //以爆炸的风格关闭$("#myExplode2")的提示框
              hide: { effect: "explode",delay: 500 } });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background:cyan; }
  /*设置盒子的基本样式*/
  .myBox{ display: inline-block; width: 450px; padding: 15px; margin: 10px; background-color: #FFF;
border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; position:
relative; border-radius: 5px; }
   body { background-color: lightgray;}
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myExplode1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子
的运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作
用的数学描述。">量子力学</a>,以及海森堡的<a  id="myExplode2" href="http://baike.so.com/doc/
6631264-7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于
是否我们所观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这
个宇宙的概念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立
大学研究人员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测
就是一切,而现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myExplode1").tooltip({ hide:{ effect:"explode",delay:500 } })表示在鼠标指针离开工具提示框 500 毫秒之后以爆炸的风格关闭该工具提示框。需要说明的是，tooltip()方法是工具提示框部件(Tooltip Widget)的方法，因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA148.html。

365 以卷帘式风格关闭和显示超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 slideUp 和 slideDown，从而实现以卷帘方式显示和关闭超链接的工具提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将以卷帘方式在两秒内从上向下展开工具提示框，效果如图 365-1 所示；当鼠标指针离开超链接 500 毫秒之后，则将以卷帘方式在两秒内从下向上折叠该工具提示框，效果如图 365-2 所示。有关此实例的主要代码如下。

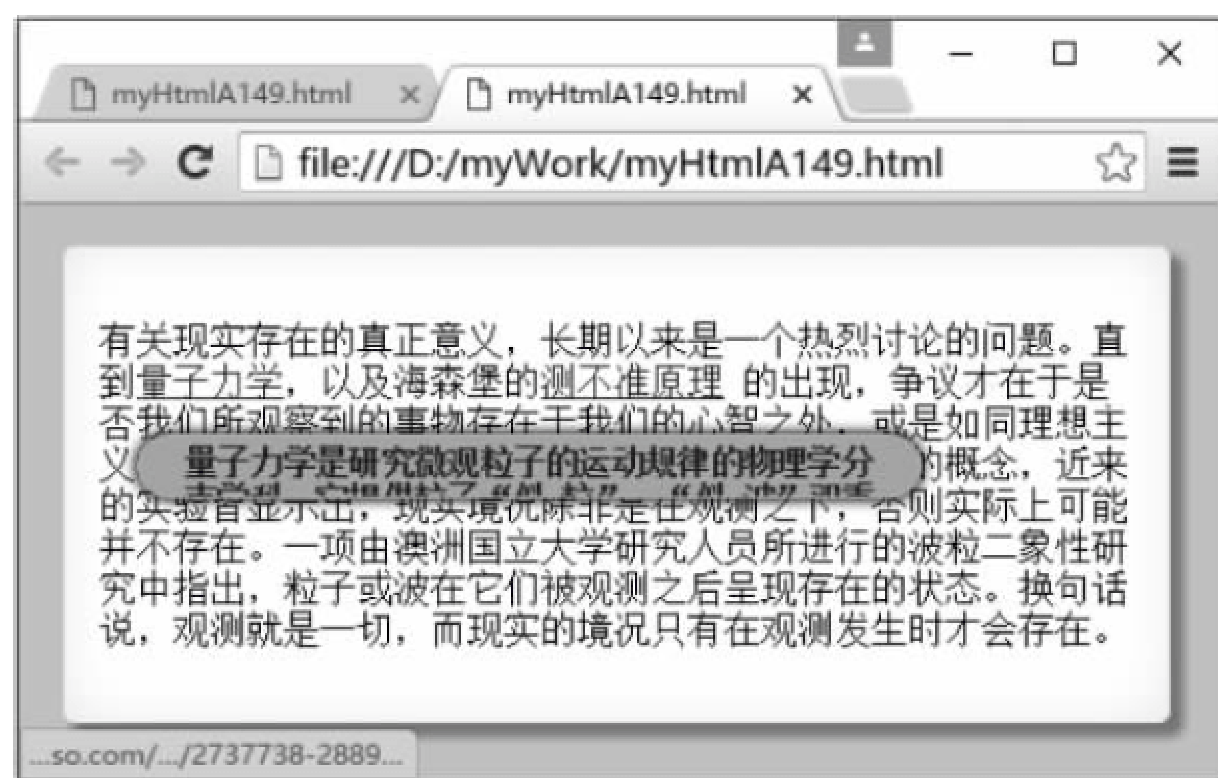


图 365-1

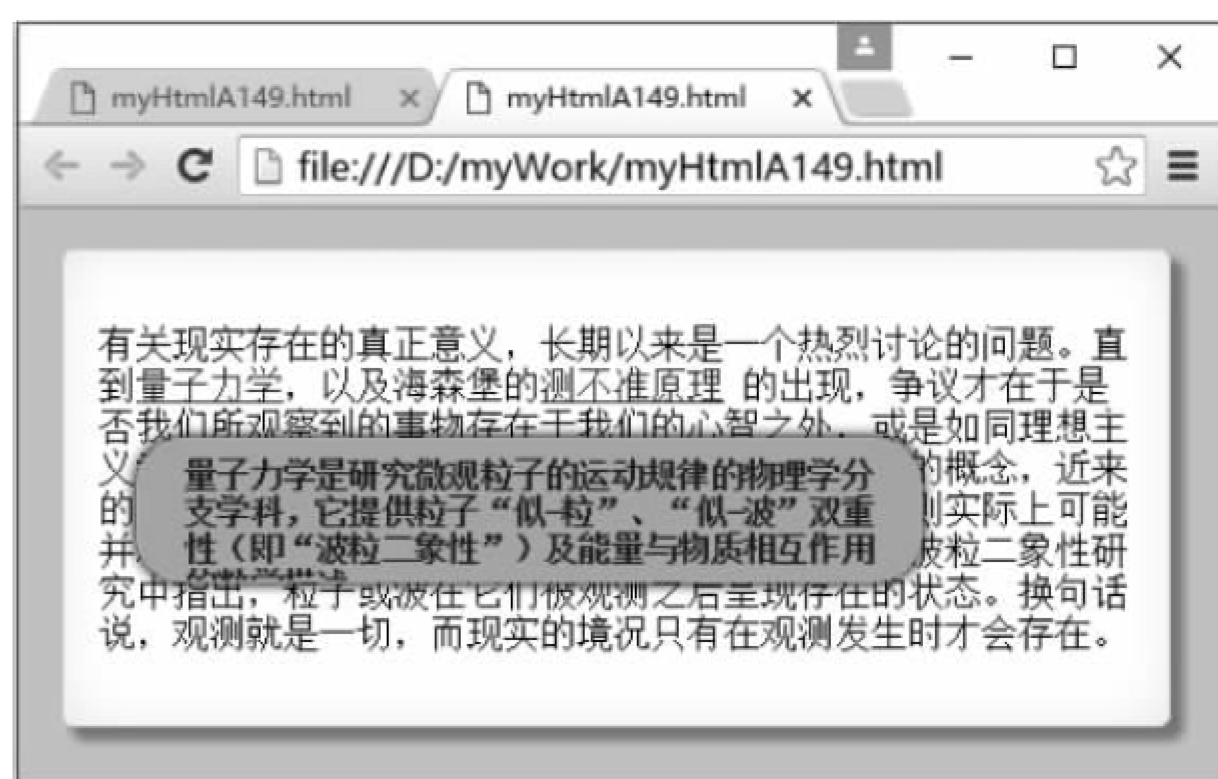


图 365-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
```

```
    $ (function() {
     $ ( "#mySlide1,#mySlide2 ").tooltip({//以卷帘式的风格关闭超链接的提示框
       hide: { effect: "slideUp", delay:500,duration:2000 } });
     $ ( "#mySlide1,#mySlide2").tooltip({ //以卷帘式的风格显示超链接的提示框
       show: { effect: "slideDown", delay:500,duration:2000 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif;box-shadow: 0 0 7px black; background:lightgreen; }
  /*设置盒子的基本样式*/
  .myBox{ display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0,
0, 0, 0.06) inset; position: relative; border-radius: 5px; }
  body { background-color: lightgray; }
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"mySlide1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a  id="mySlide2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#mySlide1,#mySlide2 ").tooltip({ hide:{ effect:"slideUp",delay:500,duration:2000 }})表示鼠标指针离开工具提示框500毫秒之后在两秒内以卷帘方式从下向上折叠该工具提示框,其中,hide表示该特效在关闭工具提示框的时候产生作用;effect:"slideUp"表示特效是向上折叠;delay:500表示延迟时间是500毫秒;duration:2000表示特效的持续时间是2000毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA149.html。

366 以滑入、滑出效果显示和关闭超链接提示框

此实例主要在jquery-ui.min.js和jquery-ui.min.css的tooltip()方法中设置特效参数drop,从而实现以滑入、滑出方式显示和关闭超链接提示框。当在Google Chrome浏览器中显示该页面时,如果将鼠标指针悬浮在"量子力学"超链接上,则将从左向右滑入工具提示框,如图366-1所示;当鼠标指针离开超链接250毫秒之后,则将从右向左在2500毫秒内滑出工具提示框,如图366-2所示。注意,在动画未执行完之前工具提示框是半透明的。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $ (function() {
    $ ("#mySlide1,#mySlide2 ").tooltip({//以滑入方式显示超链接的提示框
```

```
            show: { effect: "drop",delay: 250,duration:2500 } });
    $("#mySlide1,#mySlide2").tooltip({//以滑出方式关闭超链接的提示框
            hide: { effect: "drop",delay: 250, duration:2500 } });});
  </script>
  <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   /*设置提示框的基本样式*/
   .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: lightseagreen; }
   /*设置盒子的基本样式*/
   .myBox { display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0,
0, 0, 0.06) inset; position: relative; border-radius: 5px; }
   body { background-color: lightgray;}
  </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"mySlide1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a id="mySlide2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```



图 366-1

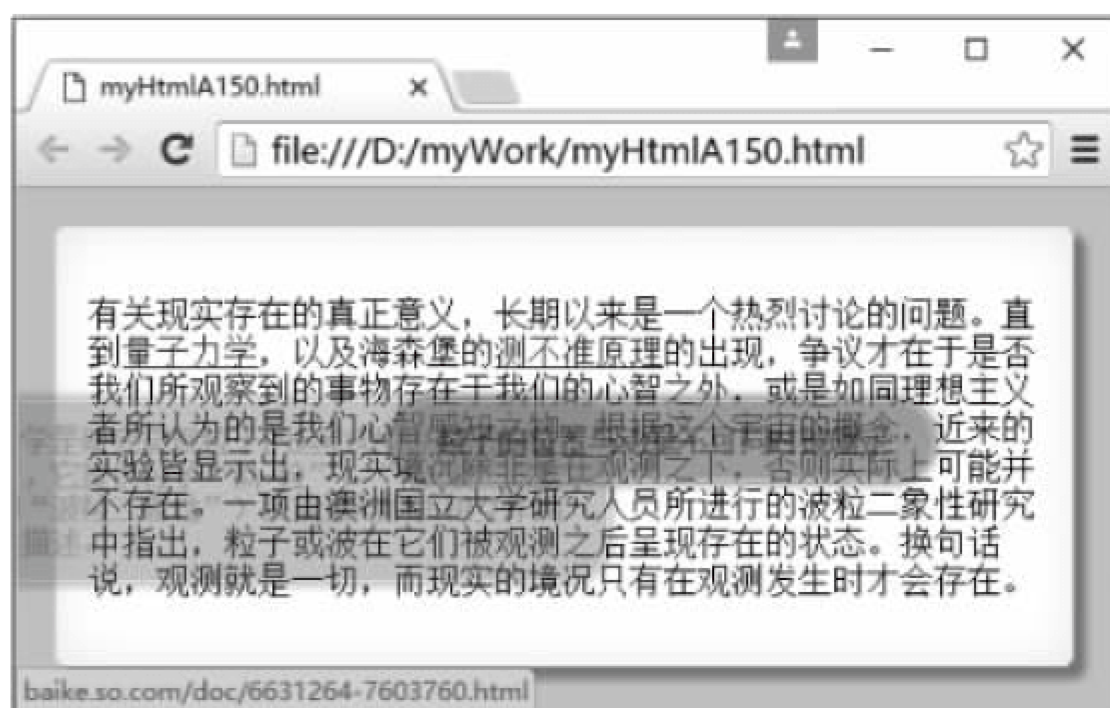


图 366-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#mySlide1,#mySlide2").tooltip({ show:{ effect:"drop",delay:250,duration:2500 } })表示鼠标指针悬浮于超链接 250 毫秒之后在 2500 毫秒内从左向右滑入工具提示框，其中，show 表示该特效在显示工具提示框的时候产生作用；effect:"drop"表示特效是抽屉风格；delay:250 表示延迟时间是 250 毫秒；duration:2500 表示特效的持续时间是 2500 毫秒。需要说明的是，tooltip()方法是工具提示框部件(Tooltip Widget)的方法，因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA150.html。

367　以扩张的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 clip，从而实现以从中间向上、下两端扩张的方式来显示超链接的提示框，或者实现以从上、下两端向中间折叠的方式来关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将从中间向上、下两端扩张以显示超链接的提示框，如图 367-1 所示；当鼠标指针离开超链接 250 毫秒之后，则将从上、下两端向中间折叠以关闭超链接的提示框，如图 367-2 所示。有关此实例的主要代码如下。

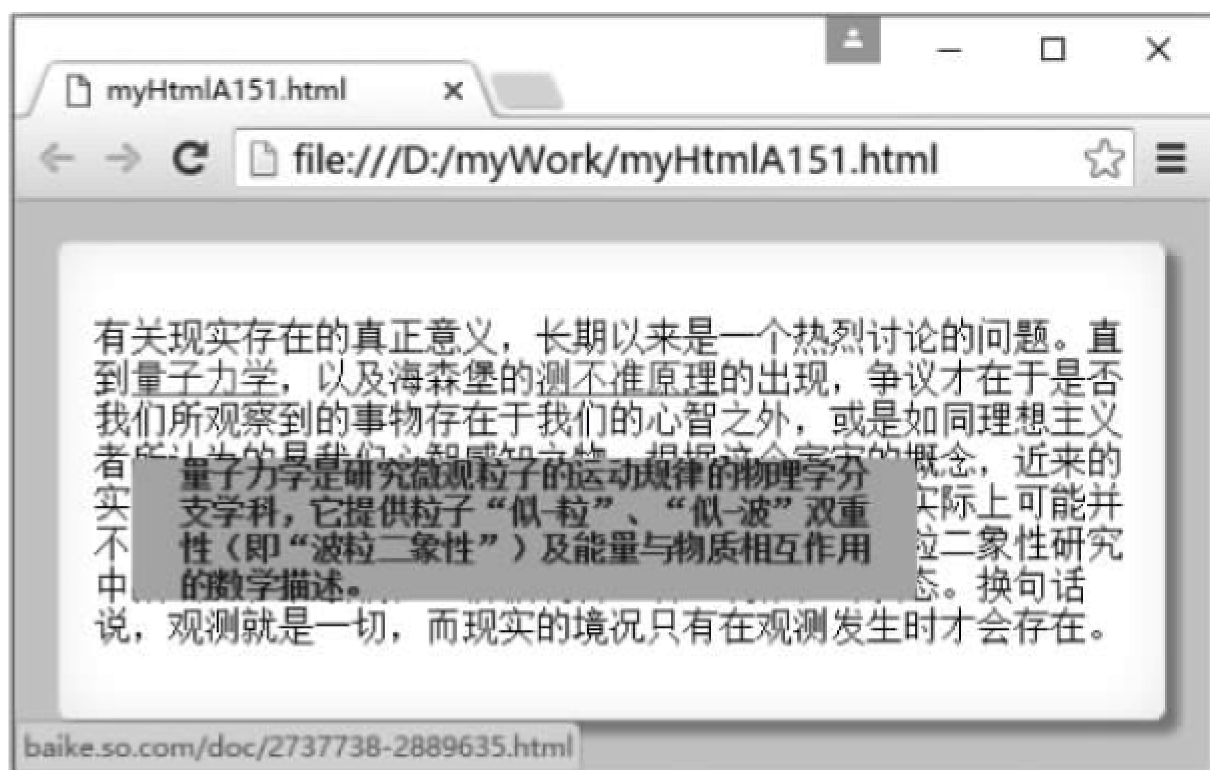


图　367-1

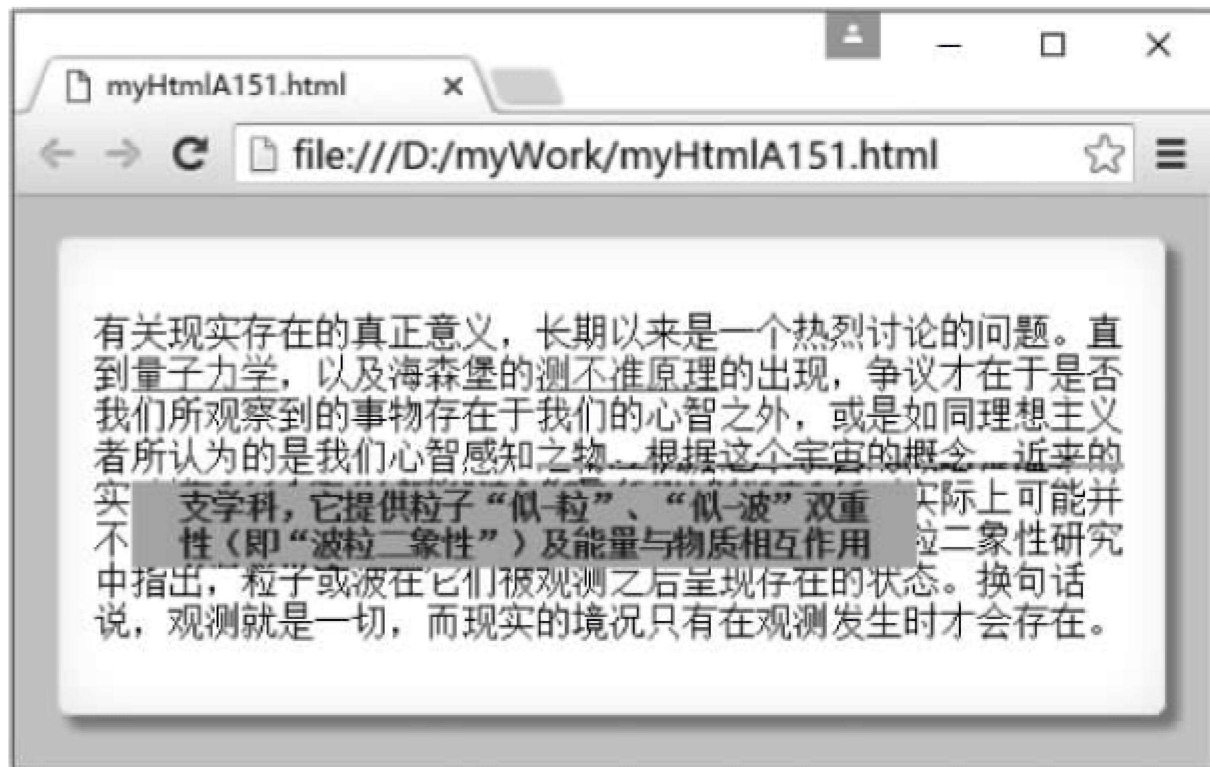


图　367-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myClip1,#myClip2").tooltip({//从中间向上、下两端扩张从而显示超链接的提示框
                show: { effect: "clip", delay: 250, duration:2500 } });
   $("#myClip1,#myClip2").tooltip({//从上、下两端向中间折叠从而关闭超链接的提示框
                hide: { effect: "clip",delay: 250,duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
 <style type="text/css">
   /*设置提示框的基本样式*/
   .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: lightskyblue; }
   /*设置盒子的基本样式*/
   .myBox { display: inline-block; width: 450px; padding: 15px; margin: 10px;  background-color:
#FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
position: relative;  border-radius: 5px; }
   body { background-color: lightgray; }
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myClip1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的运
动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用的
数学描述。">量子力学</a>,以及海森堡的<a id="myClip2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myClip1,#myClip2").tooltip({ show:{ effect:"clip",delay:250, duration:2500 } })表示鼠标指针悬浮于超链接 250 毫秒之后在 2500 毫秒内从中间向上、下两端扩张以显示超链接的提示框,其中,show 表示该特效在显示工具提示框的时候产生作用;effect:" clip"表示特效是扩张(或为 hide 时折叠)风格;delay:250 表示延迟时间是 250 毫秒;duration:2500 表示特效的持续时间是 2500 毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA151.html。

368 以淡入、淡出效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 fade,从而实现以淡入的效果显示超链接的提示框,或者以淡出的效果关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在“量子力学”超链接上,则将以淡入的效果显示超链接的提示框,如图 368-1 所示;当鼠标指针离开超链接 250 毫秒之后,则将以淡出的效果关闭超链接的提示框,如图 368-2 所示。有关此实例的主要代码如下。



图 368-1



图 368-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myFade1,#myFade2").tooltip({//以淡入的效果显示超链接的提示框
               show: { effect: "fade",delay: 250,duration:2500 } });
   $("#myFade1,#myFade2").tooltip({//以淡出的效果关闭超链接的提示框
               hide: { effect: "fade",delay: 250, duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
 <style type="text/css">
   /*设置提示框的基本样式*/
   .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: lightgoldenrodyellow; }
   /*设置盒子的基本样式*/
   .myBox { display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0,
0, 0, 0.06) inset; position: relative; border-radius: 5px; }
   body { background-color: lightgray; }
 </style></head>
```

```
<body><div class = "myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id =
"myFade1" href = "http://baike.so.com/doc/2737738 - 2889635.html" title = "量子力学是研究微观粒子的运
动规律的物理学分支学科,它提供粒子"似 - 粒"、"似 - 波"双重性(即"波粒二象性")及能量与物质相互作用的
数学描述。">量子力学</a>,以及海森堡的<a id = "myFade2" href = "http://baike.so.com/doc/6631264 -
7603760.html" title = "粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myFade1,#myFade2").tooltip({show:{effect:"fade",delay:250,duration:2500 }})表示鼠标指针悬浮于超链接 250 毫秒之后在 2500 毫秒内以淡入的效果显示超链接的提示框,其中,show 表示该特效在显示工具提示框的时候产生作用;effect:"fade"表示特效是淡入(或为 hide 时淡出)风格;delay:250 表示延迟时间是 250 毫秒;duration:2500 表示特效的持续时间是 2500 毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA152.html。

369 以膨胀的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 puff,从而实现以膨胀的效果显示和关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在"量子力学"超链接上,则将以膨胀的效果显示超链接的提示框,如图 369-1 所示;当鼠标指针离开超链接 250 毫秒之后,则将以与膨胀相反的效果关闭超链接的提示框,如图 369-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myPuff1,#myPuff2").tooltip({//以膨胀的效果显示超链接的提示框
              show: { effect: "puff",delay: 250, duration:2500 } });
   $("#myPuff1,#myPuff2").tooltip({//以与膨胀相反的效果关闭超链接的提示框
              hide: { effect: "puff",delay: 250, duration:2500 } });});
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css">
 <style type = "text/css">
  /* 设置提示框的基本样式 */
  .ui - tooltip { padding: 10px 20px; border - radius: 20px; font: bold 14px "Helvetica Neue", Sans -
Serif; box - shadow: 0 0 7px black; background: lightgoldenrodyellow; }
  /* 设置盒子的基本样式 */
  .myBox { display: inline - block; width: 450px;padding: 15px;margin: 10px;
background - color: #FFF;border: 1px solid #EEE; box - shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0,
0, 0.06) inset; position: relative; border - radius: 5px;}
  body { background - color: lightgray; }
 </style></head>
```

```
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id="myPuff1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用的数学描述。">量子力学</a>,以及海森堡的<a id="myPuff2" href="http://baike.so.com/doc/6631264-7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myPuff1,#myPuff2").tooltip({ show:{ effect:"puff",delay:250,duration:2500 }})表示鼠标指针悬浮于超链接250毫秒之后在2500毫秒内以膨胀的效果显示超链接的提示框,其中,show表示该特效在显示工具提示框的时候产生作用;effect:"puff"表示特效是膨胀(或为hide时与膨胀相反)风格;delay:250表示延迟时间是250毫秒,duration:2500表示特效的持续时间是2500毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA153.html。

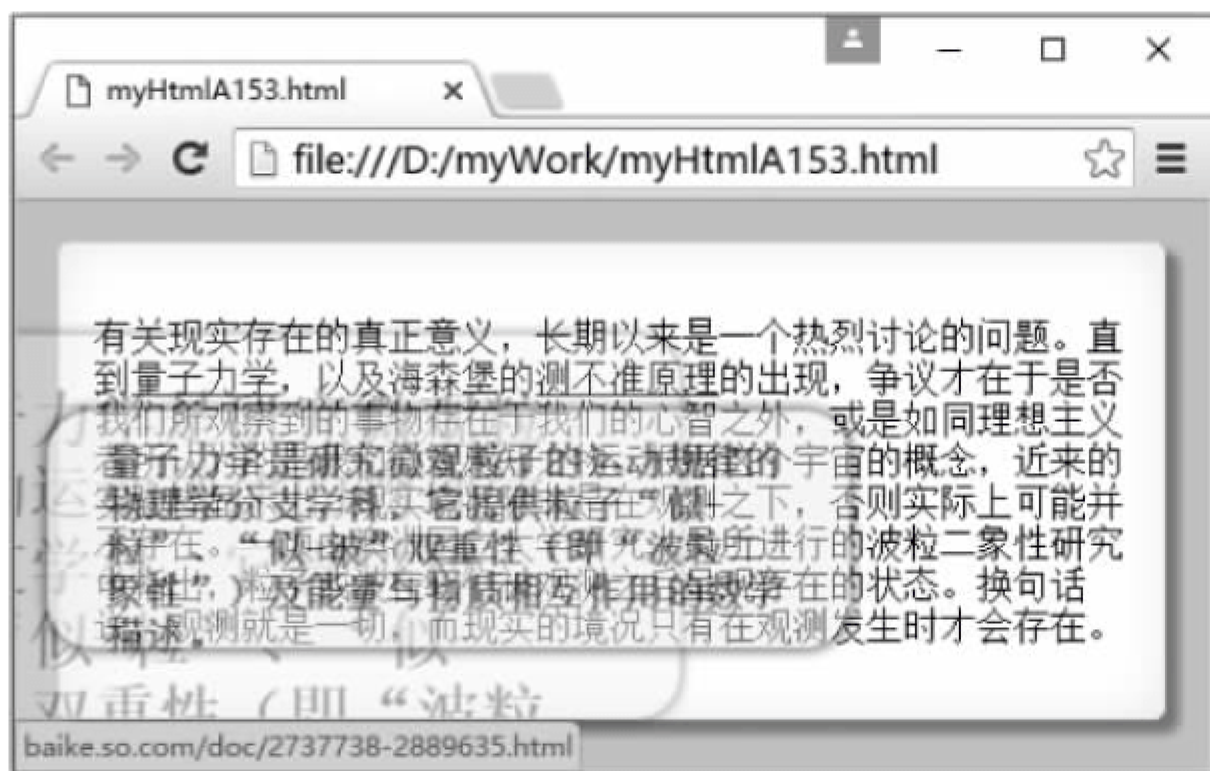


图　369-1



图　369-2

370 以闪烁的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 pulsate，从而实现以闪烁的效果显示和关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将以闪烁的效果显示超链接的提示框，如图 370-1 所示；当鼠标指针离开超链接 250 毫秒之后，将以闪烁的效果关闭超链接的提示框，如图 370-2 所示。有关此实例的主要代码如下。

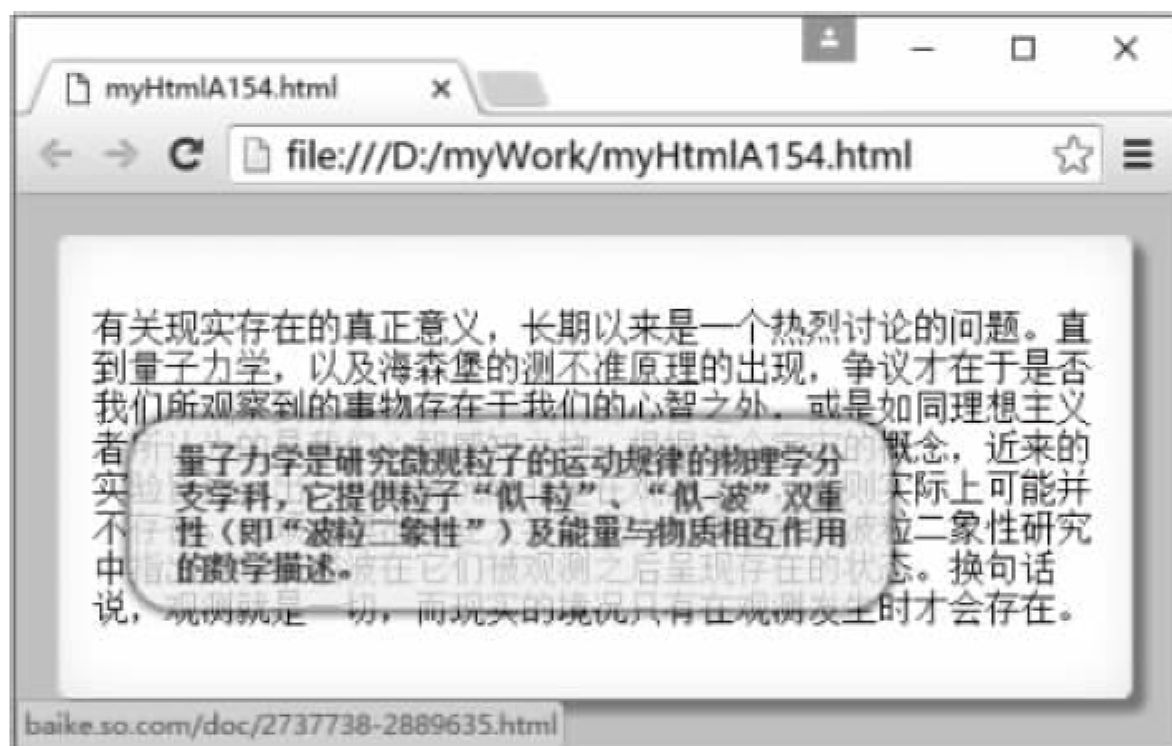


图 370-1



图 370-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myPulsate1,#myPulsate2").tooltip({//以闪烁(脉动)的效果显示超链接的提示框
                show: { effect: "pulsate",delay: 250, duration:2500 } });
   $("#myPulsate1,#myPulsate2").tooltip({//以闪烁(脉动)的效果关闭超链接的提示框
                hide: { effect: "pulsate",delay: 250, duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
 <style type="text/css">
```

```
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-Serif; box-shadow: 0 0 7px black; background: lightgoldenrodyellow; }
  /*设置盒子的基本样式*/
  .myBox {  display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; position: relative;  border-radius: 5px; }
  body { background-color: lightgray; }
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id="myPulsate1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用的数学描述。">量子力学</a>,以及海森堡的<a id="myPulsate2" href="http://baike.so.com/doc/6631264-7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myPulsate1, #myPulsate2").tooltip({ show: { effect: "pulsate",delay: 250,duration:2500 }})表示鼠标指针悬浮于超链接 250 毫秒之后在 2500 毫秒内以闪烁的效果显示超链接的提示框,其中,show 表示该特效在显示工具提示框的时候产生作用;effect: "pulsate"表示特效是闪烁(为 hide 时相同)风格;delay:250 表示延迟时间是 250 毫秒;duration:2500 表示特效的持续时间是 2500 毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA154.html。

371　以缩放的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 scale,从而实现以从中心向四周扩展的效果来显示超链接的提示框,和以从四周向中心收缩的效果来关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在“量子力学”超链接上,则将以从中心向四周扩展的效果来显示超链接的提示框,如图 371-1 所示;当鼠标指针离开超链接 250 毫秒之后,则以从四周向中心收缩的效果来关闭超链接的提示框,如图 371-2 所示。有关此实例的主要代码如下。

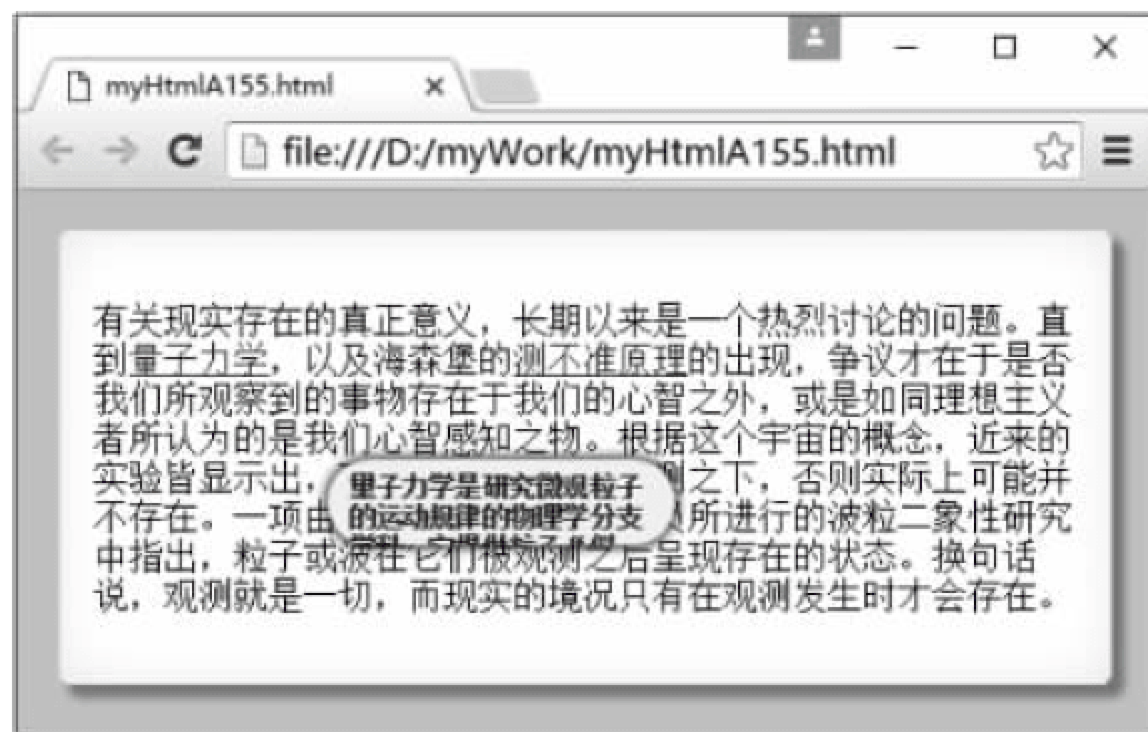


图　371-1



图 371-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
     $("#myScale1, #myScale2").tooltip({//以从中心向四周扩展来显示超链接的提示框
                   show: { effect: "scale",delay: 250, duration:2500 } });
    $("#myScale1, #myScale2").tooltip({//以从四周向中心收缩来关闭超链接的提示框
                   hide: { effect: "scale",delay: 250, duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif;box-shadow: 0 0 7px black; background: lightgoldenrodyellow; }
  /*设置盒子的基本样式*/
  .myBox { display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0,
0, 0, 0.06) inset; position: relative;border-radius: 5px; }
  body { background-color: lightgray;}
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myScale1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a id="myScale2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myScale1, #myScale2").tooltip({ show: { effect: "scale",delay: 250, duration:2500 } })表示鼠标指针悬浮于超链接250毫秒之后在2500毫秒内以从中心向四周扩展的效果来显示超链接的提示框,其中,show表示该特效在显示工具提示框的时候产生作用;effect: "scale"表示特效是缩放(为hide时方向相反)风格;delay: 250表示延迟时间是250毫秒;duration:2500表示特效的持续时间是2500毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA155.html。

372　以弹跳的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 bounce，从而实现以弹跳的效果显示和关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将以篮球落地的弹跳效果显示超链接的提示框，如图 372-1 所示；当鼠标指针离开超链接 250 毫秒之后，则将以反向的篮球落地的弹跳效果关闭超链接的提示框，如图 372-2 所示。有关此实例的主要代码如下。

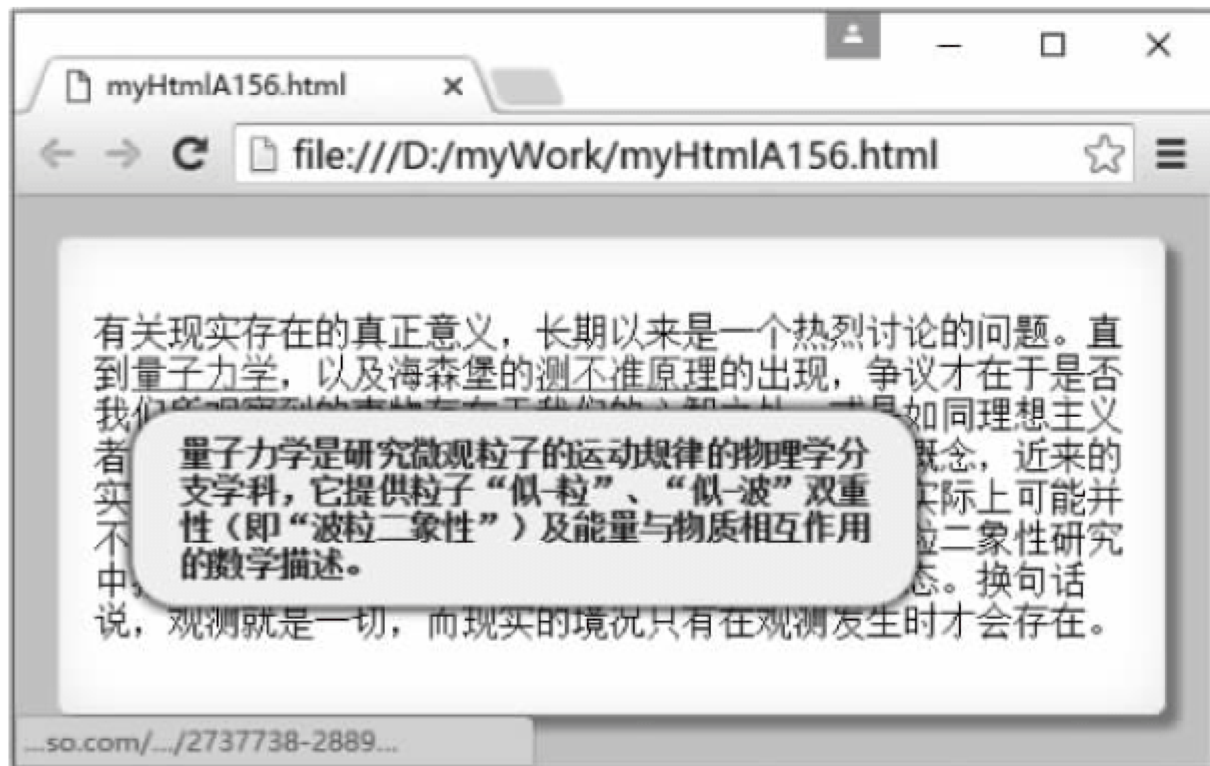


图　372-1

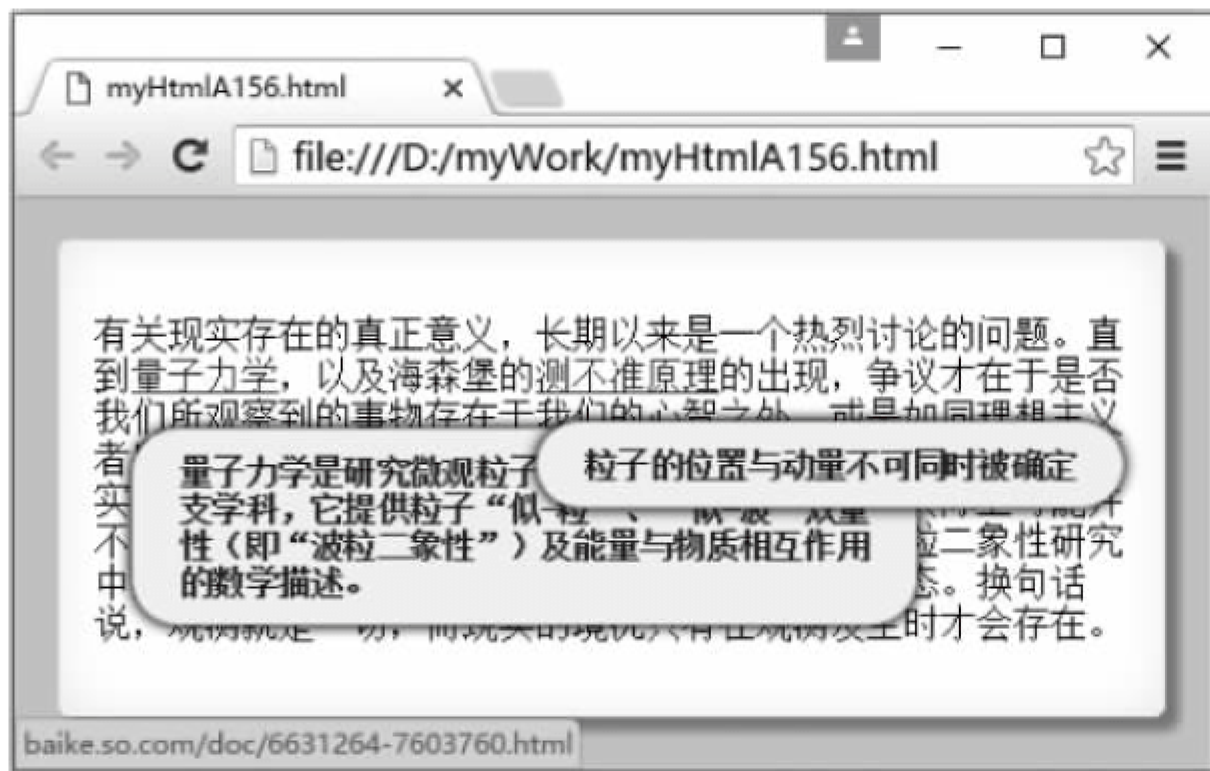


图　372-2

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
   $ (function() {
    $ ("#myBounce1, #myBounce2").tooltip({//以弹跳的效果显示超链接的提示框
                show: { effect: "bounce",delay: 250,duration:2500 } });
    $ ("#myBounce1, #myBounce2").tooltip({//以反向的弹跳效果关闭超链接的提示框
                hide: { effect: "bounce",delay: 250,duration:2500 } });});
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
```

```
    /*设置提示框的基本样式*/
    .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: lightgoldenrodyellow; }
    /*设置盒子的基本样式*/
    .myBox {  display: inline-block; width: 450px; padding: 15px; margin: 10px;
background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0,
0, 0, 0.06) inset;  position: relative; border-radius: 5px; }
    body { background-color: lightgray; }
  </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myBounce1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a id="myBounce2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myBounce1, #myBounce2").tooltip({show:{effect:"bounce",delay:250, duration:2500 } })表示鼠标指针悬浮于超链接 250 毫秒之后在 2500 毫秒内以篮球落地的弹跳效果来显示超链接的提示框,其中,show 表示该特效在显示工具提示框的时候产生作用;effect:"bounce"表示特效是弹跳(为 hide 时方向相反)风格;delay:250 表示延迟时间是 250 毫秒;duration:2500 表示特效的持续时间是 2500 毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA156.html。

373 以高亮的效果显示和关闭超链接提示框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法中设置特效参数 highlight,从而实现以高亮的效果显示和关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时,如果将鼠标指针悬浮在“量子力学”超链接上,则将以高亮效果显示超链接的提示框,如图 373-1 所示;当鼠标指针离开超链接 250 毫秒之后,则将以反向的高亮效果关闭超链接的提示框,如图 373-2 所示。有关此实例的主要代码如下。

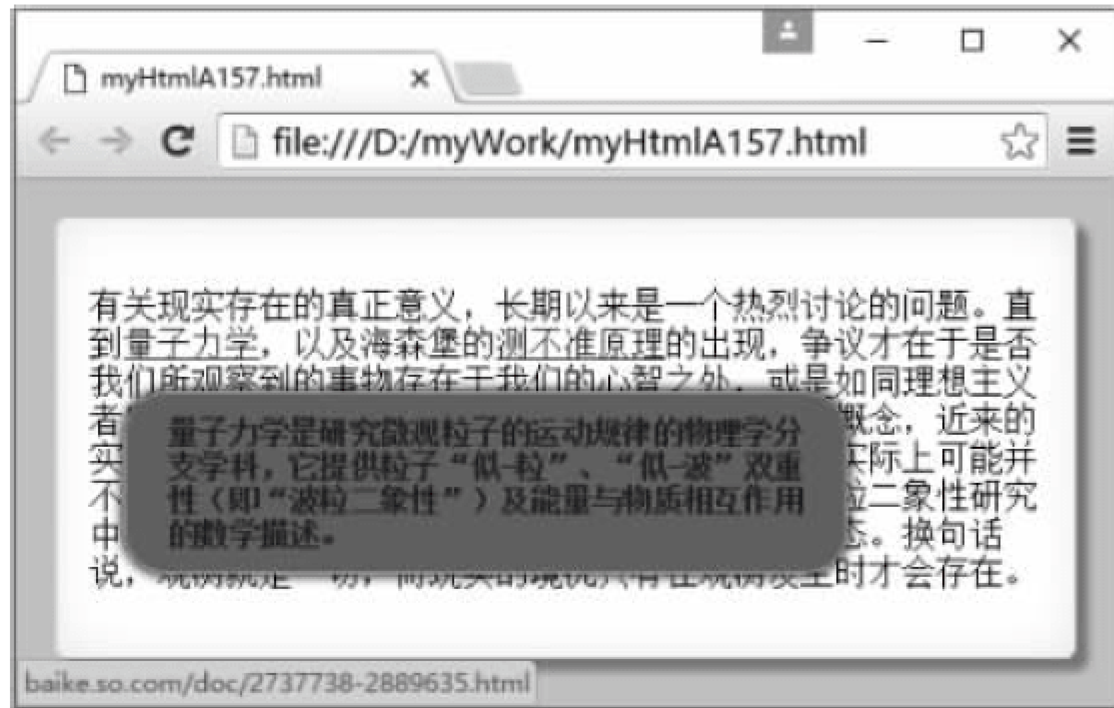


图 373-1



图　373-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myHighlight1,#myHighlight2").tooltip({//以高亮的效果来显示超链接的提示框
              show: { effect: "highlight",delay: 250,duration:2500 } });
   $("#myHighlight1,#myHighlight2").tooltip({//以高亮的反向效果关闭超链接的提示框
              hide: { effect: "highlight",delay: 250,duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: green; }
  /*设置盒子的基本样式*/
  .myBox { display: inline-block; width: 450px; padding: 15px; margin: 10px;
       background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0
       40px rgba(0, 0, 0, 0.06) inset; position: relative; border-radius: 5px;}
  body { background-color: lightgray; }
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myHighlight1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒
子的运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互
作用的数学描述。">量子力学</a>,以及海森堡的<a id="myHighlight2" href="http://baike.so.com/doc/
6631264-7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于
是否我们所观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这
个宇宙的概念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立
大学研究人员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测
就是一切,而现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myHighlight1,#myHighlight2").tooltip({show:{ effect:"highlight",delay:250,duration:2500 } })表示鼠标指针悬浮于超链接250毫秒之后在2500毫秒内以高亮效果显示超链接的提示框，其中，show表示该特效在显示工具提示框的时候产生作用；effect:"highlight"表示特效是高亮(为hide时方向相反)风格；delay:250表示延迟时间是250毫秒；duration:2500表示特效的持续时间是2500毫秒。需要说明的是，tooltip()方法是工具提示框部件(Tooltip Widget)的方法，因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA157.html。

374 以震动的效果显示和关闭超链接提示框

此实例主要在 jquery-ui. min. js 和 jquery-ui. min. css 的 tooltip()方法中设置特效参数 shake，从而实现以震动的效果显示和关闭超链接的提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则将在水平方向上以左右来回震动的效果显示超链接的提示框，如图 374-1 所示；当鼠标指针离开超链接 250 毫秒之后，则将以反向的震动效果关闭超链接的提示框，如图 374-2 所示。有关此实例的主要代码如下。

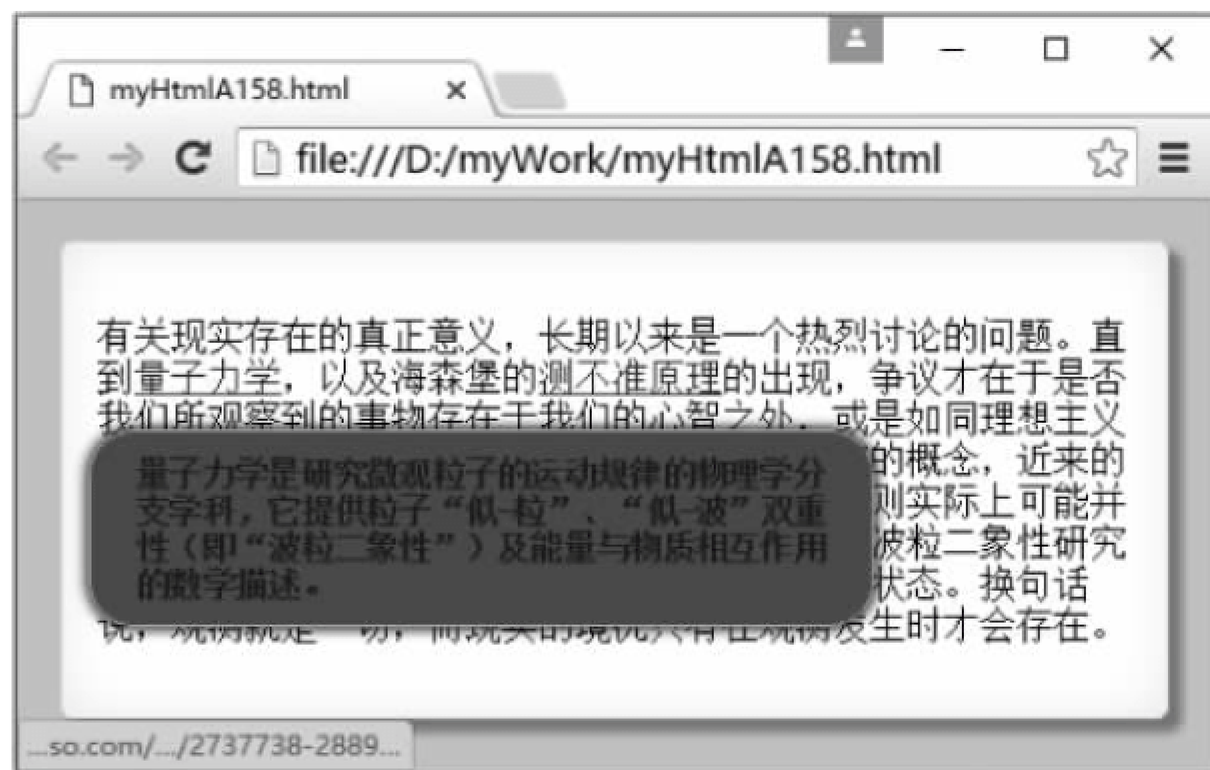


图 374-1

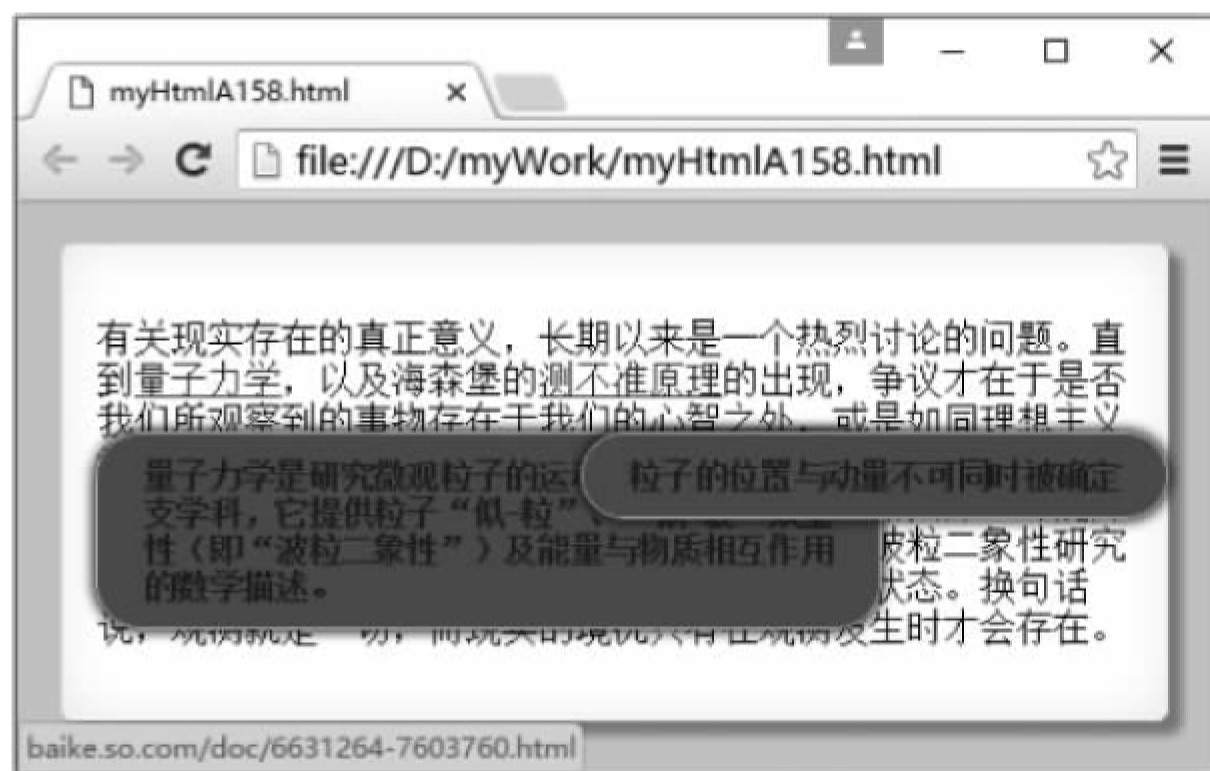


图 374-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myShake1,#myShake2").tooltip({//以震动的效果显示超链接的提示框
              show: { effect: "shake",delay: 250, duration:2500 } });
   $("#myShake1,#myShake2").tooltip({//以震动的反向效果关闭超链接的提示框
              hide: { effect: "shake",delay: 250, duration:2500 } }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
```

```
    /*设置提示框的基本样式*/
    .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
Serif; box-shadow: 0 0 7px black; background: green; }
    /*设置盒子的基本样式*/
    .myBox { display: inline-block; width: 450px;padding: 15px; margin: 10px; background-color: #FFF;
            border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06)
            inset; position: relative; border-radius: 5px; }
    body { background-color: lightgray;}
  </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"myShake1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a id="myShake2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myShake1, #myShake2").tooltip({show:{effect:"shake",delay:250, duration:2500 } })表示鼠标指针悬浮于超链接250毫秒之后在2500毫秒内在水平方向上以左右来回震动的效果显示超链接的提示框,其中,show表示该特效在显示工具提示框的时候产生作用;effect: "shake"表示特效是震动(为hide时方向相反)风格;delay:250表示延迟时间是250毫秒;duration:2500表示特效的持续时间是2500毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA158.html。

375 以抽屉滑动的效果显示和关闭超链接提示框

此实例主要在jquery-ui.min.js和jquery-ui.min.css的tooltip()方法中设置特效参数slide,从而实现以抽屉滑动的效果显示和关闭超链接的提示框。当在Google Chrome浏览器中显示该页面时,如果将鼠标指针悬浮在“量子力学”超链接上,则将在水平方向上从左至右滑出超链接的提示框,如图375-1所示;当鼠标指针离开超链接250毫秒之后,则将反向滑动关闭超链接的提示框,如图375-2所示。有关此实例的主要代码如下。

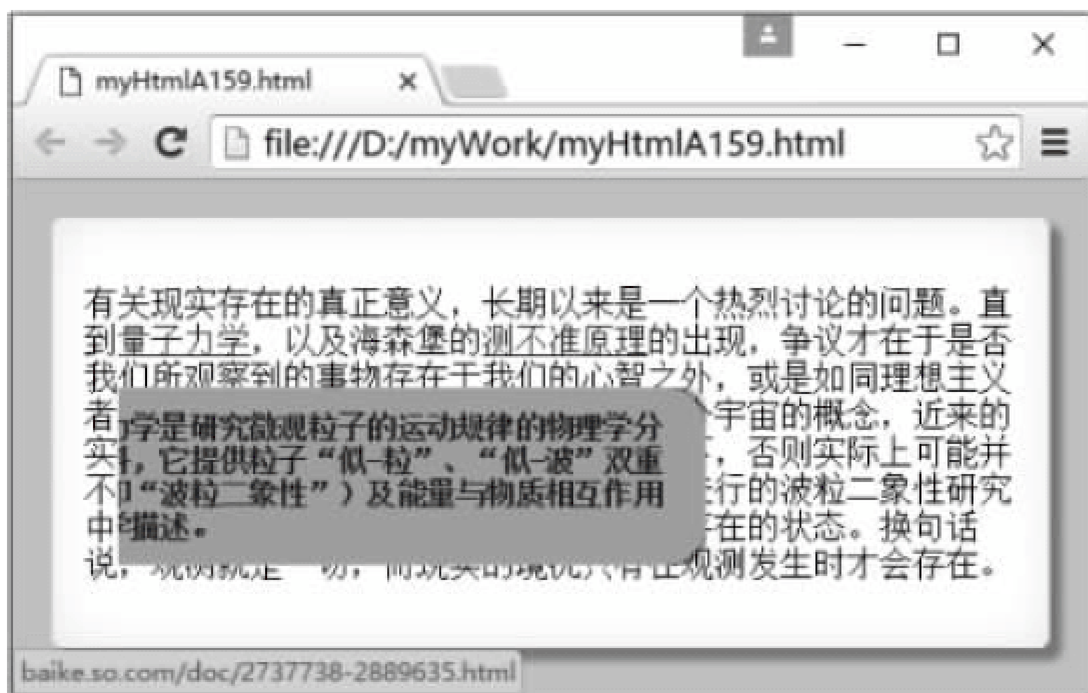


图 375-1



图 375-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#mySlide1,#mySlide2").tooltip({//以抽屉滑动的效果来显示超链接的提示框
              show: { effect: "slide",delay: 250,duration:2500 } });
   $("#mySlide1,#mySlide2").tooltip({//以抽屉滑动的反向效果来关闭超链接的提示框
              hide: { effect: "slide",delay: 250, duration:2500 } });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /*设置提示框的基本样式*/
  .ui-tooltip { padding: 10px 20px; border-radius: 20px; font: bold 14px "Helvetica Neue", Sans-
              Serif; box-shadow: 0 0 7px black; background: cyan;}
  /*设置盒子的基本样式*/
  .myBox { display: inline-block; width: 450px;padding: 15px;margin: 10px; background-color: #FFF;
         border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06)
         inset; position: relative;border-radius: 5px;}
  body { background-color: lightgray;}
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a id=
"mySlide1" href="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的
运动规律的物理学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用
的数学描述。">量子力学</a>,以及海森堡的<a id="mySlide2" href="http://baike.so.com/doc/6631264-
7603760.html" title="粒子的位置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所
观察到的事物存在于我们的心智之外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概
念,近来的实验皆显示出,现实境况除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人
员所进行的波粒二象性研究中指出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而
现实的境况只有在观测发生时才会存在。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#mySlide1,#mySlide2").tooltip({show:{ effect:"slide",delay:250,duration:2500}})表示鼠标指针悬浮于超链接250毫秒之后在2500毫秒内在水平方向上从左至右滑出超链接的提示框,其中,show表示该特效在显示工具提示框的时候产生作用;effect:"slide"表示特效是滑动(为hide时方向相反)风格;delay:250表示延迟时间是250毫秒;duration:2500表示特效的持续时间是2500毫秒。需要说明的是,tooltip()方法是工具提示框部件(Tooltip Widget)的方法,因此在使用tooltip()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA159.html。

376 在显示日期选择器时附加弹跳动画效果

此实例主要在 jquery-ui. min. js 和 jquery-ui. min. css 的 datepicker ()方法中设置动画参数 datepicker("option", "showAnim", "bounce"),从而实现在显示日期选择器时附加弹跳动画效果。当在 Google Chrome 浏览器中显示该页面时,单击"出生日期:"输入框,即可用弹跳的效果显示日期选择器,如图 376-1 所示。有关此实例的主要代码如下。

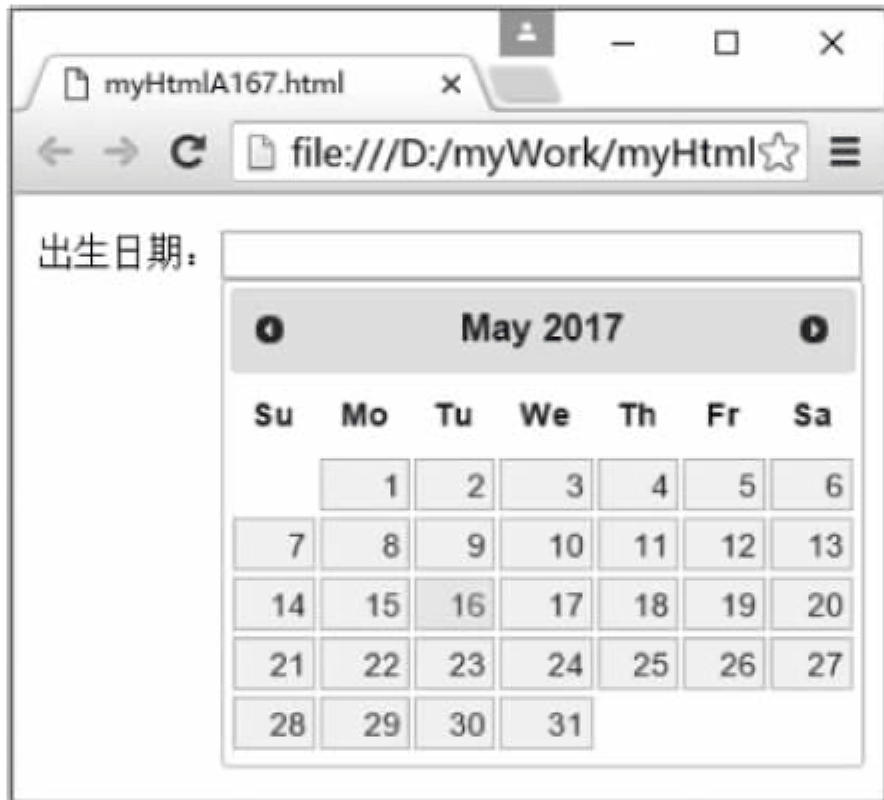


图 376-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myDatepicker").datepicker();             //显示默认的日期选择器
   //显示带弹跳效果的日期选择器
    $("#myDatepicker").datepicker("option", "showAnim","bounce" );
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   /*注意需要默认图标文件: images\ui-icons_777777_256x240.png */
   /*注意需要默认图标文件: images\ui-icons_444444_256x240.png */
   /*设置日期选择器的宽度*/
   #myDatepicker{ width: 275px;}
 </style></head>
<body><p align="center">出生日期: <input type="text" id="myDatepicker"></p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myDatepicker"). datepicker("option", "showAnim", "bounce")中的 bounce 表示弹跳效果的动画,此方法支持的其他动画有 slideDown、fadeIn、blind、clip、drop、fold、slide 等。需要说明的是,datepicker()方法是日期选择器部件(Datepicker Widget)的方法,因此在使用 datepicker ()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。此外,由于相关的图标源自默认图标文件 ui-icons_777777_256x240. png 和 ui-icons_444444_256x240. png(images 文件夹下),因此需要添加这些文件才能正确显示,否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA167. html。

第4部分 视频

377 通过拖动滑块方式改变视频的播放进度

此实例主要设置<input>标签的 type 属性为 range，从而创建一个滑块，并实现以拖动滑块的方式来改变视频的播放进度。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标向右拖动滑块，则视频的播放进度前进；如果使用鼠标向左拖动滑块，则视频的播放进度后退，如图 377-1 所示。有关此实例的主要代码如下。

图 377-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script type="text/javascript">
   $(function() {
    var myTotalMinute,myTotalSecond,myCurrentMinute,myCurrentSecond=0;
     $("#myProgress").css("width",$("video")[0].offsetWidth);
    //加载视频时获取视频的时间长度信息
     $("video")[0].addEventListener("loadedmetadata",function(){
      myTotalMinute=parseInt(this.duration/60);
      myTotalSecond=parseInt(this.duration%60);
      myCurrentMinute=0; myCurrentSecond=0;
```

```
    $ ("#myStatus").text("时间: " + myCurrentMinute + ":" + myCurrentSecond +  "/" + myTotalMinute +
":" + myTotalSecond);
    $ ("#myProgress").prop("min",0);
    $ ("#myProgress").prop("max",(this.duration).toFixed(0));
  });
  //定时同步进度条的滑块位置和视频的当前播放位置
  var timer = setInterval(function(){
   $ ("#myProgress").val( $ ("video")[0].currentTime);
  myCurrentMinute = parseInt( $ ("#myProgress").val()/60);
  myCurrentSecond = parseInt( $ ("#myProgress").val() % 60);
   $ ("#myStatus").text("时间: " + myCurrentMinute + ":" + myCurrentSecond +  "/" + myTotalMinute + ":" +
myTotalSecond);
  },10);
  //在进度条滑块被拖动时改变视频的当前播放位置
   $ ("#myProgress").on("input propertychange",function(){
   myCurrentMinute = parseInt( $ (this).val()/60);
   myCurrentSecond = parseInt( $ (this).val() % 60);
    $ ("#myStatus").text("时间: " + myCurrentMinute + ":" + myCurrentSecond +  "/" + myTotalMinute + ":"
 + myTotalSecond);
    $ ("video")[0].currentTime = parseInt( $ ("#myProgress").val());
    $ ("video")[0].play();
 });});
 </script>
 <style type = "text/css">
  body{ background - color: lightgrey; }
  video{ width:400px; height:250px; }
  input{ width:400px;}
 </style></head>
<body><div align = "center">
 <video src = "media/a073myVideo.webm" autoplay = "autoplay"></video>
 <p><input type = "range" id = "myProgress" value = "0"/></p>
 <p id = "myStatus">时间: </p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，range 属性值是 HTML5 中的<input>标签的 type 属性的新属性值，该属性值指定范围值的输入字段，在 HTML 页面中显示为滑块。$("#myProgress").prop("min",0)用于设置进度条(滑块)的最小值，$("#myProgress").prop("max",(this.duration).toFixed(0))用于根据视频的播放时间设置进度条(滑块)的最大值。prop()方法是 jQuery 的属性操作方法，用于设置或返回当前 jQuery 对象所匹配的元素的属性值，该方法的语法格式如下。

```
jQueryObject.prop( propertyName [, value ] )
```

如果指定了 value 参数，则表示设置属性 propertyName 的值为 value；如果没有指定 value 参数，则表示返回属性 propertyName 的值。

$("video")[0].currentTime 表示视频的当前播放时间位置，$("video")[0].play()表示播放视频。

此实例的源文件名是 myHtmlA176.html。

378　在播放视频、音频时同步显示关联字幕

此实例主要通过使用< track >标签实现在页面上播放视频或音频时同步显示字幕。当在浏览器中显示该页面时，单击视频工具栏上的“播放”按钮则立即播放视频，并在屏幕上同步显示字幕，如图 378-1 所示。有关此实例的主要代码如下。



图　378-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><video src = "media/a073myVideo.webm" controls width = "400px" height = "300px">
  <track kind = "subtitles" src = "media/a073myText.vtt" default /></video>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< video >标签用于定义视频，比如电影片段或其他视频流；< track >标签则用于为 video 元素之类的媒介规定外部文本轨道，即人们通常所说的字幕。< track >标签常用的属性及说明如表 378-1 所示。

表 378-1　< track >标签常用的属性及说明

属　性	值	描　述
default	default	规定该轨道是默认的，假如没有选择任何轨道
kind	captions	表示轨道属于什么文本类型
	chapters	
	descriptions	
	metadata	
	subtitles	
label	label	轨道的标签或标题
src	url	轨道的 URL
srclang	language_code	轨道的语言，若 kind 属性值是"subtitles"，则该属性是必需的

下面是此实例所采用的字幕文件 a073myText.vtt 的部分内容。

```
WEBVTT FILE
Cue - 1
```

```
00:00:00.500 --> 00:00:02.000 D:vertical A:start
The Web is always changing
2
00:00:02.500 --> 00:00:04.300
 and the way we access it is changing
3
00:00:05.000 --> 00:00:07.000
The source of that change is <c.highlight> you </c>
4
…
```

从上面的内容可以得知，track 元素是沿着 video 元素所使用的时间轴指定的时间来同步显示字幕的。track 元素所使用的字幕文件通常采用如上所示的 WebVTT 标记文件，可以在此文件中通过设置标记来自定义显示内容文字的下画线等显示风格。

此实例的源文件名是 myHtmlA073.html。

379　以一定的角度旋转正在播放的视频窗口

此实例主要使用 jQuery 的 css()方法设置 video 元素的 transform 属性，从而实现以一定的角度旋转正在播放的视频窗口。当在 Google Chrome 浏览器中显示该页面时，单击“以一定的角度旋转正在播放的视频窗口”按钮，则视频播放窗口旋转 30°之后的效果如图 379-1 所示。有关此实例的主要代码如下。



图　379-1

```
<!doctype html><html><head><meta charset = UTF - 8 >
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function rotateVideo(){                    //以一定的角度旋转正在播放的视频窗口
   $("#myVideo").css("position","relative");
   $("#myVideo").css("top","55px");     //设置窗口左上角的 top 坐标
```

```
    $("#myVideo").css("left","35px");              //设置窗口左上角的 left 坐标
    $("#myVideo").css("transform","rotate(30deg)");
  }
 </script></head>
<body><p><input type="button" value="以一定的角度旋转正在播放的视频窗口"
onclick="rotateVideo()" style="width:400px"></p>
<video id="myVideo" src="media/myMovie.ogg" controls="controls" width="330px">
</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的一个或多个样式属性。jQuery 代码 css("transform", "rotate(30deg)")等效于 CSS 代码 style="transform: rotate(30deg)"，即把 video 元素旋转 30°。

此实例的源文件名是 myHtmlA101.html。

380 在垂直方向上翻转正在播放的视频图像

此实例主要使用 jQuery 的 css()方法设置 video 元素的 transform 属性，从而实现在垂直方向上翻转正在播放的视频图像。当在 Google Chrome 浏览器中显示该页面时，单击"垂直翻转视频图像"按钮，则正在播放的视频画面在垂直方向上翻转之后的效果如图 380-1 所示。有关此实例的主要代码如下。



图 380-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function flipVideo(){//垂直翻转视频图像
    $("#myVideo").css("transform","scaleY(-1)");
  }
 </script></head>
<body><p><input type="button" value="垂直翻转视频图像" onclick="flipVideo()"
style="width:350px"></p>
<video id="myVideo" src="media/myMovie.ogg" controls="controls" width="350px">
 当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个jQuery方法，用于设置或返回被选元素的一个或多个样式属性，该方法的语法声明如下。

```
$(selector).css(name,value)
```

其中，参数name用于规定CSS属性的名称，该参数可包含任何CSS属性，例如"color"；参数value是可选参数，用于规定CSS属性的值，该参数可包含任何CSS属性值，例如"red"。如果设置了空字符串值，则从元素中删除指定属性。

jQuery代码css("transform","scaleY(－1)")等效于CSS代码style="transform:scaleY(－1)"，即在垂直方向上翻转选择的元素。

此实例的源文件名是myHtmlA089.html。

381 在水平方向上镜像正在播放的视频图像

此实例主要使用jQuery的css()方法设置video元素的transform属性，从而实现在水平方向上镜像正在播放的视频图像。当在Google Chrome浏览器中显示该页面时，单击"水平镜像视频图像"按钮，则正在播放的视频画面在水平方向上镜像之后的效果如图381-1所示。有关此实例的主要代码如下。



图 381-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function mirrorVideo(){//水平镜像视频图像
   $("#myVideo").css("transform","scaleX(-1)");
  }
 </script></head>
<body><p><input type="button" value="水平镜像视频图像" onclick="mirrorVideo()" style="width:
350px"></p>
<video id="myVideo" src="media/myMovie.ogg" controls="controls" width="350px">
 当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个jQuery方法，用于

设置或返回被选元素的一个或多个样式属性。jQuery 代码 css("transform", "scaleX(－1)")等效于 CSS 代码 style="transform:scaleX(－1)",即在水平方向上镜像选择的元素。

此实例的源文件名是 myHtmlA090.html。

382 在垂直方向上压缩视频产生宽屏幕效果

此实例主要使用 jQuery 的 css()方法设置 video 元素的 transform 属性,从而实现在垂直方向上压缩视频图像产生宽屏幕效果。当在 Google Chrome 浏览器中显示该页面时,单击"在垂直方向上压缩视频图像产生宽屏幕效果"按钮,则正在播放的视频图像在垂直方向上压缩之后的宽屏幕效果如图 382-1 所示。有关此实例的主要代码如下。



图 382-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function wideVideo(){//在垂直方向上压缩视频图像产生宽屏幕效果
   $("#myVideo").css("transform","scaleY(0.75)");
   //在水平方向上压缩视频图像
   //$("#myVideo").css("transform","scaleX(0.75)");
   //在水平和垂直方向上同时压缩视频图像
   //$("#myVideo").css("transform","scale(0.75)");
  }
 </script></head>
<body><p><input type="button" value="在垂直方向上压缩视频图像产生宽屏幕效果"
onclick="wideVideo()" style="width:350px"></p>
<video id="myVideo" src="media/myMovie.ogg" controls="controls" width="350px">
 当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,css()方法是一个 jQuery 方法,用于设置或返回被选元素的一个或多个样式属性。jQuery 代码 css("transform", "scaleY(0.75)")等效于 CSS 代码 style="transform:scaleY(0.75)",即在垂直方向上压缩选择的元素。

此实例的源文件名是 myHtmlA091.html。

383　访问摄像头并单击截取视频中的图像

此实例主要通过使用 navigator.getUserMedia()方法和 drawImage()方法实现访问本地视频摄像头并单击截取图像。在测试实例程序时首先打开本地视频摄像头设备，然后在 Google Chrome 浏览器中显示该页面，此时将自动同步显示本地视频摄像头设备录制的视频，如图 383-1 的上半部分所示，单击视频画面，将截取该视频画面，如图 383-1 的下半部分所示。有关此实例的主要代码如下。



图　383-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   navigator.getUserMedia = navigator.getUserMedia ||
            navigator.webkitGetUserMedia || window.navigator.mozGetUserMedia;
    window.URL = window.URL || window.webkitURL;
    var myVideo = document.getElementById('myVideo');
    var myCanvas = document.getElementById('myCanvas');
    var myContext = myCanvas.getContext('2d');
    var myMedia = null;
    //访问本地摄像头
    navigator.getUserMedia({video:true, audio:false},
     function(stream) {
      myVideo.src = window.URL.createObjectURL(stream);
      myMedia = stream;
     },function(err) {
      alert(err);
     });
     $("#myVideo").click(function() {//单击视频画面截取图像
      if (myMedia) {
```

```
        myContext.drawImage(myVideo, 0, 0);
        document.getElementById('myImage').src = myCanvas.toDataURL('image/png');
  } });});
 </script></head>
<body><video id="myVideo" width="400" height="300" autoplay></video>
<img src="" id="myImage" />
<canvas width="400" height="300" style="display:none;" id="myCanvas">
</canvas></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，navigator.getUserMedia()方法用于访问客户端摄像头或麦克风等视频/音频输入设备，该方法使用3个参数。第1个参数值为一个约束(constraints)对象，该对象具有一个video属性及一个audio属性，属性值均为布尔类型，当video属性值为true时代表捕捉视频信息，当video属性值为false时代表不捕捉视频信息；当audio属性值为true时代表捕捉音频信息，当audio属性值为false时代表不捕捉音频信息。第2个参数值是开发者自行定义的当访问用户设备成功时所执行的回调函数，该回调函数具有一个参数，参数值是一个MediaStream对象，在浏览器执行getUserMedia()方法时将自动创建该对象，该对象代表同步媒体数据流。第3个参数值是开发者自行定义的当访问用户本地设备失败时所执行的回调函数，该回调函数具有一个参数，参数值是一个error对象，代表浏览器抛出的错误对象。drawImage()方法用于将video元素的某一帧画面输出到canvas元素。

此实例的源文件名是myHtmlA088.html。

384 使用article元素嵌入文件来播放视频

此实例主要通过使用article元素实现以插件的形式来播放视频等媒体文件。当在浏览器中显示该页面时将自动播放预置的视频文件，如图384-1所示。有关此实例的主要代码如下。



图 384-1

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><article><object>
  <param name="allowFullScreen" value="true">
   <embed src="media/myMovie.ogg" width="350" height="200"></embed>
  </param></object></article>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<article>标签是HTML5新增的标签。<article>标签定义外部的内容，外部内容可以是来自一个外部的新闻提供者的一篇新的文章，或

者来自 blog 的文本，或者是来自论坛的文本，亦或是来自其他外部源内容。除此之外，也可以像实例这样在其中嵌入视频文件播放视频。

此实例的源文件名是 myHtmlA020. html。

385　在页面中播放 mp4 等格式的多媒体文件

此实例主要通过使用< video >标签实现在页面中播放 mp4 等格式的多媒体文件。当在浏览器中显示该页面时预置的视频处于等待状态，单击控制栏上的“播放”按钮，则将播放视频，如图 385-1 所示。有关此实例的主要代码如下。

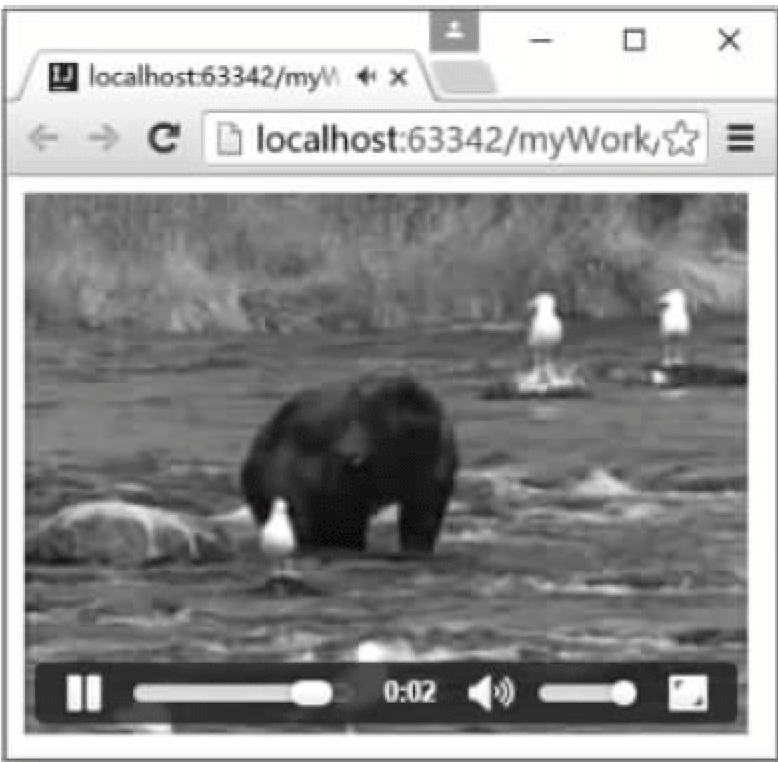


图　385-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><video src = "media/myMovie.ogg" controls = "controls" width = "350px" >
 当前浏览器不支持此视频播放功能</video>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< video >标签是 HTML5 新增的用于播放多媒体文件的标签，该标签对应的 video 元素支持 Ogg、MPEG4、WebM 等 3 种视频格式，其中 Ogg 是带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件，MPEG4 是带有 H. 264 视频编码和 AAC 音频编码的 MPEG4 文件，WebM 是带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件。video 元素的 control 属性提供用于控制媒体的播放、暂停和音量等按钮，src 属性指定多媒体文件的 URL。video 元素常用的属性及说明如表 385-1 所示。

表 385-1　video 元素常用的属性

属　　性	值	描　　述
autoplay	autoplay	如果出现该属性，则视频在就绪后马上播放
controls	controls	如果出现该属性，则向用户显示控件，比如“播放”按钮
height	pixels	设置视频播放器的高度
loop	loop	如果出现该属性，则当媒介文件完成播放后再次开始播放
preload	preload	如果出现该属性，则视频在页面加载时进行加载并预备播放；如果使用 "autoplay"，则忽略该属性
src	url	要播放的视频的URL
width	pixels	设置视频播放器的宽度

注意：<video>与</video>之间插入的内容是供不支持 video 元素的浏览器显示的。

此实例的源文件名是 myHtmlA017.html。

386 使用滤镜改变视频图像的颜色透明度

此实例主要通过在 jQuery 的 css()方法中采用滤镜来实现以滤镜的方式改变视频图像的颜色透明度。当在 Google Chrome 浏览器中显示该页面时，单击“使用滤镜改变视频图像的颜色透明度”按钮，则正在播放的视频图像的颜色透明度将发生改变，如图 386-1 所示。有关此实例的主要代码如下。



图 386-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function opacityVideo(){//使用滤镜改变视频图像的颜色透明度
    $("#myVideo").css(" - webkit - filter","opacity(0.3)"); }
 </script></head>
<body><p><input type = "button" value = "使用滤镜改变视频图像的颜色透明度"
onclick = "opacityVideo()" style = "width:350px"></p>
<video id = "myVideo" src = "media/myMovie.ogg" controls = "controls" width = "350px">当前浏览器不支持此
视频播放功能</video>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的属性。jQuery 代码 css("-webkit-filter","opacity(0.3)")等效于 CSS 代码 style="-webkit-filter: opacity(0.3)"。opacity(0.3)的参数值在 0 到 1 之间，0 表示完全透明，1 表示完全不透明。

此实例的源文件名是 myHtmlA100.html。

387 通过滤镜使视频图像产生怀旧的风格

此实例主要使用 jQuery 的 css()方法进行颜色过滤，从而使视频图像产生怀旧风格的特效。当在 Google Chrome 浏览器中显示该页面时，单击“采用滤镜使视频图像产生怀旧风格的特效”按钮，则正在播放的视频图像产生的怀旧风格的效果如图 387-1 所示。有关此实例的主要代码如下。



图　387-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function sepiaVideo(){//采用滤镜使视频图像产生怀旧风格的特效
   $ ("#myVideo").css(" - webkit - filter","sepia(80 % )");
 }
 </script></head>
<body><p><input type = "button" value = "采用滤镜使视频图像产生怀旧风格的特效"
onclick = "sepiaVideo()" style = "width:350px"></p>
 <video id = "myVideo" src = "media/myMovie.ogg" controls = "controls" width = "350px">  当前浏览器不支
持此视频播放功能</video>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的属性。jQuery 代码 css("-webkit-filter","sepia(80%)")等效于 CSS 代码 style="-webkit-filter:sepia(80%)"，sepia(80%)方法中的百分比值越大效果越明显。

此实例的源文件名是 myHtmlA092.html。

388　通过色相旋转使视频图像显示重组色彩

此实例主要使用 jQuery 的 css()方法改变图像的色相，从而使视频图像呈现改变色相之后的重组色彩。当在 Google Chrome 浏览器中显示该页面时，单击"通过色相旋转使视频图像显示重组色彩"按钮，则正在播放的视频图像将呈现重组色彩之后的特殊效果，如图 388-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function rotateVideo(){//通过色相旋转使视频图像显示重组色彩
   $ ("#myVideo").css(" - webkit - filter","hue - rotate(120deg)");
  }
 </script></head>
<body><p>
 <input type = "button" value = "通过色相旋转使视频图像显示重组色彩"
```

```
onclick = "rotateVideo()" style = "width:350px"></p>
  <video id = "myVideo" src = "media/myMovie.ogg" controls = "controls" width = "350px">当前浏览器不支持
此视频播放功能</video>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "hue-rotate(120deg)")等效于 CSS 代码 style="-webkit-filter: hue-rotate(120deg)"。

此实例的源文件名是 myHtmlA099.html。



图 388-1

389 通过设置饱和度改变(淡化)视频图像的颜色

此实例主要使用 jQuery 的 css()方法设置饱和度，从而淡化视频图像的颜色。当在 Google Chrome 浏览器中显示该页面时，单击“设置饱和度淡化视频图像的颜色”按钮，则正在播放的视频图像的颜色淡化之后的效果如图 389-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  function saturateVideo(){ //设置饱和度淡化视频图像的颜色
   $("#myVideo").css(" - webkit - filter","saturate(50 %)");  }
 </script></head>
<body><p><input type = "button" value = "设置饱和度淡化视频图像的颜色"
onclick = "saturateVideo()" style = "width:350px"></p>
  <video id = "myVideo" src = "media/myMovie.ogg" controls = "controls" width = "350px">当前浏览器不支持
此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "saturate(50%)")等效于 CSS 代码 style="-webkit-filter: saturate(50%)"，saturate(50%)方法中的百分比值越小淡化效果越明显，saturate(0%)是黑白效果，saturate(700%)表示颜色超浓。

此实例的源文件名是 myHtmlA093.html。



图　389-1

390　反转颜色使视频图像产生底片的特效

此实例主要使用 jQuery 的 css()方法反转颜色，从而使视频图像产生底片的特效。当在 Google Chrome 浏览器中显示该页面时，单击“通过反转颜色使视频图像产生底片的特效”按钮，则正在播放的视频图像的颜色将以底片的效果显示，如图 390-1 所示。有关此实例的主要代码如下。



图　390-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function invertVideo(){//通过反转颜色使视频图像产生底片的特效
   $("#myVideo").css("-webkit-filter","invert(100%)");
  }
```

```
</script></head>
<body><p><input type="button" value="通过反转颜色使视频图像产生底片的特效"
onclick="invertVideo()" style="width:350px"></p>
  <video id="myVideo" src="media/myMovie.ogg" controls="controls" width="350px"> 当前浏览器不支
持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "invert(100%)")等效于 CSS 代码 style="-webkit-filter:invert(100%)"。invert(100%)用于反转当前的颜色。

此实例的源文件名是 myHtmlA094.html。

391 改变对比度突出显示视频图像的颜色

此实例主要使用 jQuery 的 css()方法改变对比度，从而突出显示视频图像的颜色。当在 Google Chrome 浏览器中显示该页面时，单击"改变对比度突出显示视频图像的颜色"按钮，则正在播放的视频图像的颜色对比度将增强，效果如图 391-1 所示。有关此实例的主要代码如下。



图 391-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function contrastVideo(){//改变对比度突出显示视频图像的颜色
   $("#myVideo").css("-webkit-filter","contrast(200%)");
  }
 </script></head>
<body><p><input type="button" value="改变对比度突出显示视频图像的颜色"
onclick="contrastVideo()" style="width:350px"></p>
  <video id="myVideo" src="media/myMovie.ogg" controls="controls"
width="350px"> 当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "contrast(200%)")等效于 CSS 代码 style="-webkit-filter: contrast(200%)"，contrast(200%)方法中的百分比值越大颜色对比效果

越强烈。

此实例的源文件名是 myHtmlA095.html。

392　改变明亮度提高视频图像的呈现颜色

此实例主要使用 jQuery 的 css()方法改变明亮度，从而提高视频图像的呈现颜色。当在 Google Chrome 浏览器中显示该页面时，单击"改变明亮度提高视频图像的呈现颜色"按钮，则正在播放的视频图像的颜色明亮度将提高，效果如图 392-1 所示。有关此实例的主要代码如下。



图　392-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function brightnessVideo(){//改变明亮度提高视频图像的呈现颜色
    $("#myVideo").css("-webkit-filter","brightness(200%)");
  }
 </script></head>
<body><p><input type="button" value="改变明亮度提高视频图像的呈现颜色"
onclick="brightnessVideo()" style="width:350px"></p>
  <video id="myVideo" src="media/myMovie.ogg" controls="controls"
width="350px">当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "brightness(200%)")等效于 CSS 代码 style="-webkit-filter: brightness(200%)"，brightness(200%)方法中的百分比值越大颜色明亮度越高。

此实例的源文件名是 myHtmlA096.html。

393　改变模糊度使视频图像呈现虚化特效

此实例主要使用 jQuery 的 css()方法改变模糊度，从而使视频图像呈现虚化特效。当在 Google Chrome 浏览器中显示该页面时，单击"改变模糊度使视频图像呈现虚化特效"按钮，则正在播放的视

频图像将呈现虚化特效,效果如图 393-1 所示。有关此实例的主要代码如下。



图 393-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function blurVideo(){//改变模糊度使视频图像呈现虚化特效
   $("#myVideo").css("-webkit-filter","blur(2px)");
  }
 </script></head>
<body><p><input type="button" value="改变模糊度使视频图像呈现虚化特效"
onclick="blurVideo()" style="width:350px"></p>
  <video id="myVideo" src="media/myMovie.ogg" controls="controls"
width="350px">当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,css()方法是一个 jQuery 方法,用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter", "blur(2px)")等效于 CSS 代码 style="-webkit-filter:blur(2px)",blur(2px)方法中的参数值越大虚化程度越高。

此实例的源文件名是 myHtmlA097.html。

394 改变灰度使视频图像呈现黑白效果

此实例主要使用 jQuery 的 css()方法改变灰度等级,从而使视频图像呈现黑白效果。当在 Google Chrome 浏览器中显示该页面时,单击"改变灰度等级使视频图像呈现黑白效果"按钮,则正在播放的视频图像将呈现黑白效果,如图 394-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
  <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   function grayscaleVideo(){//改变灰度等级使视频图像呈现黑白效果
    $("#myVideo").css("-webkit-filter","grayscale(100%)");
   }
 </script></head>
<body><p><input type="button" value="改变灰度等级使视频图像呈现黑白效果"
```

```
onclick = "grayscaleVideo()" style = "width:350px"></p>
  <video id = "myVideo" src = "media/myMovie.ogg" controls = "controls"
width = "350px">当前浏览器不支持此视频播放功能</video></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，css()方法是一个 jQuery 方法，用于设置或返回被选元素的样式属性。jQuery 代码 css("-webkit-filter"，"grayscale(100%)")等效于 CSS 代码 style＝"-webkit-filter：grayscale(100%)"，grayscale(100%)方法中的参数值在 0% 到 100%之间，参数值越大灰度等级越高。

此实例的源文件名是 myHtmlA098.html。



图　394-1

395　全屏播放视频或退出全屏正常播放视频

此实例主要通过使用 webkitRequestFullscreen()方法实现全屏播放视频或退出全屏模式正常播放视频。当在 Google Chrome 浏览器中显示该页面时将以正常模式播放视频，如图 395-1 所示；单击"全屏播放视频"按钮将以全屏模式播放视频，在全屏模式中自动添加了"关闭"和"退出全屏模式"按钮，如图 395-2 所示；单击"退出全屏模式"按钮，则将返回正常播放模式，如图 395-1 所示。有关此实例的主要代码如下。



图　395-1



图 395-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnFullscreen").click(function() {//全屏播放视频
    if ($("video")[0].webkitRequestFullscreen) {
     $("video")[0].webkitRequestFullscreen();
  } });});
 </script>
 <style type="text/css">
  video { width: 400px; height: 250px; }
  button { width: 400px;}
 </style></head>
<body><div align="center">
 <video src="media/a073myVideo.webm" autoplay=autoplay" controls="controls"> </video><br>
 <button id="myBtnFullscreen">全屏播放视频</button></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，webkitRequestFullscreen()方法用于请求全屏模式，此方法是 Chrome 浏览器专用的方法；当前在 Firefox、Safari 等浏览器中也支持全屏请求，但它们均使用自己专用的方法，因此如果需要网页支持更多的浏览器，可以采用下面这段代码。

```
$("#myBtnFullscreen").click(function () {
 if ($("video")[0].requestFullscreen) {
    $("video")[0].requestFullscreen();
 }
 else if ($("video")[0].mozRequestFullScreen) {
    $("video")[0].mozRequestFullScreen();
 }
 else if ($("video")[0].webkitRequestFullScreen) {
    $("video")[0].webkitRequestFullScreen();
} });
```

在全屏模式中会自动添加“关闭”和“退出全屏模式”按钮，不需要使用任何额外的代码即自动实现。实际上，按下键盘上的 Esc 键也会照样退出全屏模式。exitFullscreen()方法可以使浏览器退出全屏，但是在一般情况下不会直接使用。

此实例的源文件名是 myHtmlA177.html。

第 5 部分

元素

396 以折叠或展开风格显示标题或者详细内容

此实例主要通过使用< details >标签实现单击标题即可展开或折叠对应的详细内容的功能。当在浏览器中显示该页面时将显示朱德元帅的详细内容，隐藏其他 3 位元帅的详细内容，如图 396-1 所示；单击“朱德元帅”标题，则将隐藏朱德元帅的详细内容，单击“刘伯承元帅”标题，则将显示刘伯承元帅的详细内容，如图 396-2 所示，即通过单击标题能够在隐藏详细内容或显示详细内容的两种状态之间自动切换。有关此实例的主要代码如下。



图 396-1



图 396-2

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><details open="open"><summary style="outline: none"><b>朱德元帅(1886-1976)</b>
</summary><p>1886年12月1日生,字玉阶,四川仪陇人。1909年考入云南陆军讲武堂,同年加入中国同盟会,参加了辛亥革命。1913年后在滇军任营长、副团长、团长、旅长。曾参加护国运动、护法运动...</p>
</details>
<details><summary style="outline: none"><b>刘伯承元帅(1892-1986)</b></summary>
  <p>1892年生,四川开县赵家镇人,无产阶级革命家、军事战略家、战术家、马克思主义军事理论家、军事教育家,被誉为"军神"。1912年考入重庆军政府将校学堂。1914年加入孙中山领导的中华革命党...</p>
</details>
<details><summary style="outline: none"><b>陈毅元帅(1901-1972)</b></summary>
  <p>1901年生,字仲弘,四川省乐至县人。1919年赴法国勤工俭学,1921年回国。1922年加入中国社会主义青年团。1923年加入中国共产党。1927年在武汉中央军校担任政治工作...</p></details>
<details><summary style="outline: none"><b>罗荣桓(1902-1963)</b></summary><p>1902年生,湖南省衡山(今衡东)县人。1927年加入中国共产主义青年团,同年转入中国共产党...</p></details></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<details>标签用于描述文档或文档某个部分的详细内容,该标签有一个重要属性——open,open 属性规定了在 HTML 页面上的详细内容是否可见,如果没有设置该属性,则隐藏详细内容;如果设置该属性为 open="open",则显示详细内容。

此实例的源文件名是 myHtmlA006.html。

397 以分组形式显示下拉列表框中的选项

此实例主要通过使用<optgroup>标签实现对下拉列表框内部的选项进行分组显示。当在 Google Chrome 浏览器中显示该页面时,在下拉列表框中的地级市选项将按照省级行政区进行分组,如图 397-1 所示;选择其中的"成都市"选项,则选择结果如图 397-2 所示。注意组名省级行政区是不可选择的。有关此实例的主要代码如下。



图 397-1



图 397-2

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><div>请选择送达目的地:
 <select><optgroup label = "重庆市">
    <option value = "渝北区">渝北区</option><option value = "江北区">江北区</option>
    <option value = "渝中区">渝中区</option></optgroup>
  <optgroup label = "四川省">
    <option value = "成都市">成都市</option><option value = "广元市">广元市</option>
    <option value = "绵阳市">绵阳市</option></optgroup>
  <optgroup label = "河北省">
    <option value = "石家庄市">石家庄市</option>
    <option value = "承德市">承德市</option>
    <option value = "张家口市">张家口市</option></optgroup></select></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<optgroup>标签用于定义选项组,当使用一个较长的选项列表时对相关的选项进行组合会使处理更加容易。<optgroup>标签是 HTML5 的新增标签。

此实例的源文件名是 myHtmlA112.html。

398　为下拉列表框中的选项添加关联辅助信息

此实例主要在下拉列表框的选项标签<option>中设置 label 属性值，从而实现为下拉列表框的列表选项添加辅助信息。当在 Google Chrome 浏览器中显示该页面时，单击下拉列表框右端的下拉箭头，则将弹出该下拉列表框的选项列表，如图 398-1 所示。每个选项除了前面的选项数据外，后面还有该选项的辅助信息。在下拉列表框的选项列表中任意选择一个选项，例如“http://www.baidu.com 百度搜索”，则将在下拉列表框中仅显示“http://www. baidu.com”，隐藏了该选项的辅助信息“百度搜索”，如图 398-2 所示。有关此实例的主要代码如下。

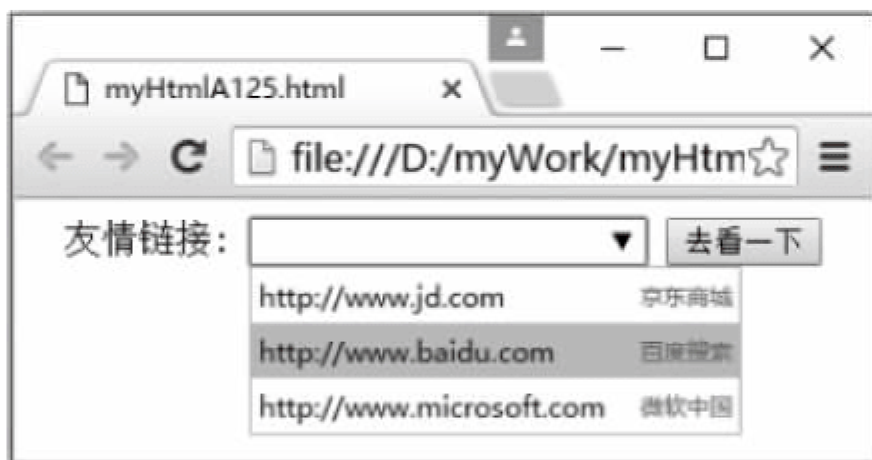


图　398-1

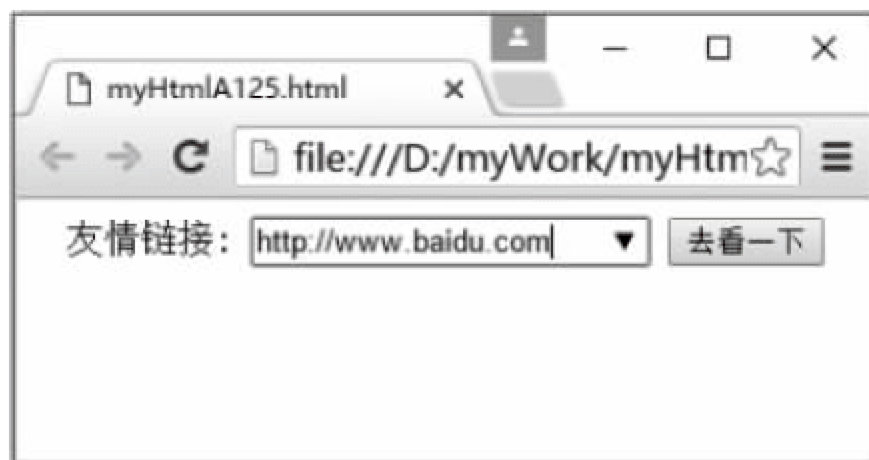


图　398-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){
   $("input[type=submit]").click(function(e){//响应鼠标单击"去看一下"按钮
    e.preventDefault();
    //跳转到当前选择的 URL 对应的页面
    location.href=$("input[type=url]").val();
  });});
 </script></head>
<body><div align="center">
 友情链接:<input type="url" list="myWebsite" />
 <datalist id="myWebsite">
  <option label="京东商城" value="http://www.jd.com" />
  <option label="百度搜索" value="http://www.baidu.com" />
  <option label="微软中国" value="http://www.microsoft.com" />
 </datalist><input type="submit" value="去看一下"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<option label="百度搜索" value="http://www.baidu.com" />中的 label 属性用于为选项规定描述，<option>用于定义下拉选项列表中的一个选项。<datalist>标签定义选项列表，它通常与 input 元素配合使用定义 input 可能的值，datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。<input>标签中的 list 属性引用<datalist>元素，其中包含<input>元素的预定义选项。

此实例的源文件名是 myHtmlA125.html。

399　使用 png 图像自定义下拉列表框的向下箭头

此实例主要通过设置-webkit-appearance 属性值为 none 来取消下拉列表框默认的实心向下箭头，然后在 background 属性中使用 png 图像，从而实现自定义下拉列表框的向下箭头。当在 Google

Chrome 浏览器中显示该页面时，下拉列表框右上角显示的即是自定义箭头，如图 399-1 所示。有关此实例的主要代码如下。

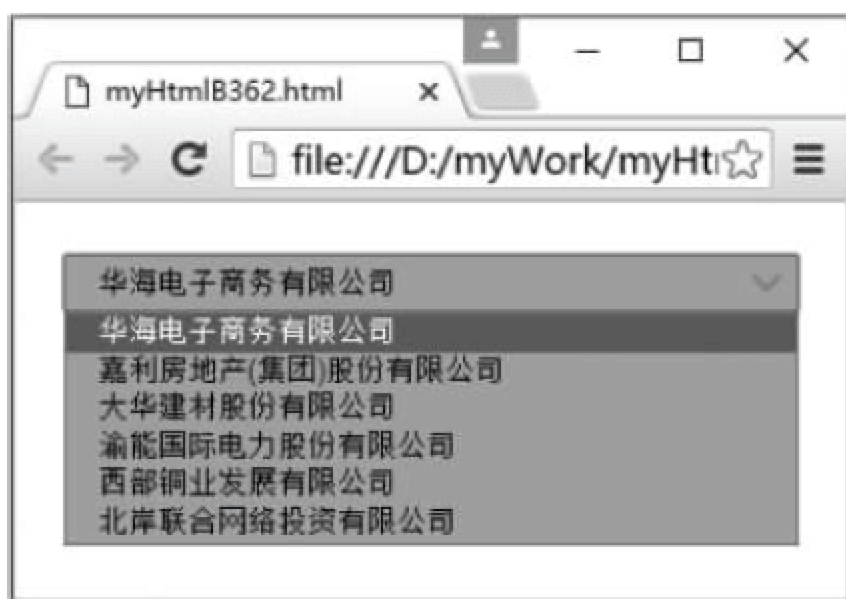


图 399-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*自定义箭头样式*/
  .myArrow{ width:320px;height:25px; -webkit-appearance:none;
            margin-top:15px;padding-right: 20px; padding-left: 10px;
            /*设置箭头图像及位置*/
            background: cyan url("img/B362.png") no-repeat 298px;
            /*设置箭头图像的大小*/
            background-size:14px 9px; }
 </style></head>
<body><div align="center"><select class="myArrow">
  <option selected>华海电子商务有限公司</option>
  <option>嘉利房地产(集团)股份有限公司</option>
  <option>大华建材股份有限公司</option>
  <option>渝能国际电力股份有限公司</option>
  <option>西部铜业发展有限公司</option>
  <option>北岸联合网络投资有限公司</option></select></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-appearance：none 用于取消下拉列表框的默认实心向下箭头，-webkit-appearance 与 CSS3 标准制定的 appearance 属性不同，appearance 属性允许使元素看上去像标准的用户界面元素，该属性据说是所有的主流浏览器都不支持。如果将-webkit-appearance：none 改为 appearance：none，默认的实心箭头仍然会出现。

此实例的源文件名是 myHtmlB362.html。

400 模拟下拉列表框的风格创建下拉式菜单

此实例主要使用 jQuery 的 slideToggle()方法控制元素的折叠和展开，从而使下拉式菜单实现类似于下拉列表框的选择风格。当在 Google Chrome 浏览器中显示该页面时，单击菜单栏右端的选项图标，则将向下展开菜单选项；当鼠标指针在各个菜单选项之间移动时，当前悬浮的菜单选项的文本将以大字号显示，如图 400-1 所示；单击需要的菜单选项，则该菜单选项的文本将显示在菜单栏中，并且整个菜单选项折叠隐藏，如图 400-2 所示。有关此实例的主要代码如下。



图 400-1



图 400-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $('nav').click(function(){                //展开或折叠下拉菜单
    $('.myItems').slideToggle("slow");
   });
   $("ul.myItems li").click(function(){      //显示选中的菜单项文本
    $("#mySelected").text($(this).find("a").text());
 });});
 </script>
 <style type="text/css">
  body { background-color: #313233;font-family: arial, san-serif; }
  /*设置导航菜单栏的样式*/
  nav { width: 350px; margin:0 auto; margin-top:50px; }
  /*设置菜单栏盒子的样式*/
  .myBox { background-color: green; border-radius: 10px; color: #FFF; display: block; position:
         relative; font-size: 20px; padding:20px; -webkit-transition: 0.3s ease; border-bottom:
         3px solid #CC6025; }
  /*设置右端菜单图标盒子的样式*/
  .myIcon { display: block; width:24px; margin-top:2px; float:right; }
  /*设置右端菜单图标每条横线的样式*/
  .myIcon span { width:24px; height:4px;border-radius:3px; color:#FFF;
              display:block; background: white; margin:2px }
  /*设置鼠标指针悬浮菜单栏时的样式*/
  nav:hover .myBox{ cursor:pointer; background: lightgreen; }
  /*设置下拉菜单项盒子的基本样式*/
  .myItems{background: lightpink; border-radius: 10px;text-align: center; margin: 0; padding:0; list
         -style:none; position: relative;margin-top:20px; display:none;}
  /*设置下拉菜单盒子的三角形箭头*/
```

```
.myItems:before { content: ""; position:absolute; border: 15px solid lightpink;
                 border-top: 0 solid transparent! important;border-right: 15px solid transparent!
                 important;border-left:15px solid transparent! important; right:10px;top: -15px }
/*设置菜单项的基本样式*/
.myItems a { text-decoration: none; display:block;
            padding:20px 0; font-weight:700; font-size:16px }
/*设置偶数行菜单项的背景颜色*/
.myItems li:nth-child(even) a { background:cyan; }
/*设置在鼠标指针悬浮时的菜单项样式*/
.myItems li:hover a { font-size: 24px; color:red; }
</style></head>
<body><nav><div class="myBox">
 <span id="mySelected">选择您最喜欢的书籍...</span>
 <div class="myIcon"><span></span><span></span><span></span></div></div>
 <ul class="myItems"><li><a href="#">HTML5 从入门到精通</a></li>
  <li><a href="#">HTML5 经典实例</a></li>
  <li><a href="#">HTML5 移动 Web 开发指南</a></li>
  <li><a href="#">HTML5+CSS3 从入门到精通</a></li>
  <li><a href="#">HTML5 入门经典</a></li></ul></nav></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$('.myItems').slideToggle("slow")用于折叠和展开菜单选项盒子。slideToggle()方法是一个 jQuery 方法，它通过使用滑动效果来切换元素的可见状态，如果被选元素是可见的，则隐藏这些元素，如果被选元素是隐藏的，则显示这些元素。slideToggle()方法的语法格式如下。

```
$(selector).slideToggle(speed,callback)
```

其中，参数 speed 是可选参数，它规定元素从隐藏到可见的速度(或者相反)，默认为 normal，可能的值有毫秒(比如 1500)、slow、normal、fast，在设置速度的情况下，元素在切换的过程中会逐渐地改变高度；参数 callback 也是可选参数，它表示在 slideToggle()方法执行之后将要执行的函数，除非设置了 speed 参数，否则不能设置该参数。

此实例的源文件名是 myHtmlB375.html。

401 以拖曳方式将菜单项插入新的位置

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 中的 sortable()方法实现以拖曳的方式将菜单项插入新的位置。当在 Google Chrome 浏览器中显示该页面时，默认的菜单项顺序如图 401-1 所示。使用鼠标拖动“约翰霍普金斯大学”菜单项到“圣安德鲁斯大学”菜单项和“加州理工学院”菜单项之间，如图 401-2 所示；然后松开鼠标，则“约翰霍普金斯大学”菜单项就会被插入“圣安德鲁斯大学”菜单项和“加州理工学院”菜单项之间，如图 401-3 所示。有关此实例的主要代码如下。



图 401-1

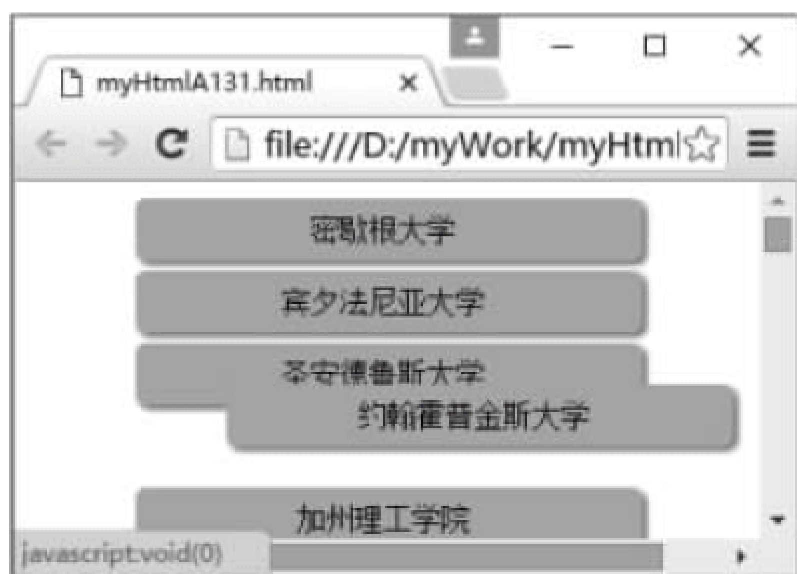


图　401-2



图　401-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#sortable").sortable({              //以拖曳的方式将菜单栏的菜单项插入新位置
    revert:true});
   $( "#sortable" ).disableSelection();   //禁用选择匹配的元素集合内的文本内容
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  .myContainer { width: 100%; height: 1500px; }  /*设置盒子的基本样式*/
  /*设置无序列表的基本样式*/
  ul { list-style-type: none; margin: 0; padding: 0; margin-bottom: 10px;}
  li { margin: 5px 0 0 0;}  /*设置列表项的基本样式*/
  /*设置列表项超链接的基本样式*/
  li a { -webkit-transition: 0.3s ease-out; background: lightgreen; color: black;
padding: 7px 15px 7px 10px; border-radius: 5px; width: 220px; display: block; text-decoration: none;
font-size: 14px; -webkit-box-shadow: 2px 2px 4px #888; }
 </style></head>
<body><div class="myContainer" align="center"><ul id="sortable">
 <li><a href="javascript:void(0)">密歇根大学</a></li>
 <li><a href="javascript:void(0)">约翰霍普金斯大学</a></li>
 <li><a href="javascript:void(0)">宾夕法尼亚大学</a></li>
 <li><a href="javascript:void(0)">圣安德鲁斯大学</a></li>
<li><a href="javascript:void(0)">加州理工学院</a></li></ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#sortable").sortable({revert:true})用于实现以拖曳的方式将菜单栏的菜单项插入新位置。sortable()方法是 jQuery UI 的可排序小部件(Sortable Widget)的方法，在任意的 DOM 元素上启用 sortable 功能，通过鼠标单击并拖曳元素到列表中的一个新的位置，其他条目会自动调整。在默认情况下，sortable 的各个条目共享 draggable 属性。需要注意的是，在使用 sortable()方法实现拖曳排序前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA131.html。

402　以拖曳方式增加和移除两个列表的选项

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 中的 sortable()方法实现以拖曳的方式增加和移除两个列表的选项。当在 Google Chrome 浏览器中显示该页面时，默认的两个教材列

表的列表项顺序如图402-1所示。使用鼠标拖动“数学基础教材”列表中的“常微分方程”列表项到“计算机专业教材”列表中的“操作系统设计与实现”和“离散数学”之间，如图402-2所示；然后松开鼠标，则“常微分方程”列表项就会被插入“操作系统设计与实现”和“离散数学”之间，如图402-3所示。反之亦然。有关此实例的主要代码如下。



图 402-1



图 402-2



图 402-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   //向 connectWith 选项传递一个选择器,把一个列表中的元素排序到另一个列表中
   $("#sortable1, #sortable2").sortable({
    connectWith: ".connectedSortable"
   }).disableSelection(); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   .myContainer { width:580px; flow: left; }/*设置盒子的基本样式*/
   /*设置无序列表的基本样式*/
   #sortable1, #sortable2 { list-style-type: none; margin: 0; padding: 0.5em 0 ;float: left; margin
                            -right: 50px; text-align: center; font-size: 24px; color: yellow; font
                            -weight: bold; background-color: darkgreen; border-radius: 5px;}
   /*设置列表项的基本样式*/
   #sortable1 li, #sortable2 li { margin: 0 5px 1px 5px;
                                  padding: 5px; font-size: 1.2em; width: 220px; }
   /*设置列表项超链接的基本样式*/
   li a { -webkit-transition: 0.3s ease-out; background: lightgreen;color: black;
         padding: 7px 15px 7px 10px; border-radius: 5px; width: 220px;display: block;
         text-decoration: none;font-size: 14px; -webkit-box-shadow: 2px 2px 4px #888; }
 </style></head>
<body><div class="myContainer">
 <ul id="sortable1" class="connectedSortable">数学基础教材
  <li><a href="javascript:void(0)">数学分析</a></li>
  <li><a href="javascript:void(0)">高等代数</a></li>
  <li><a href="javascript:void(0)">常微分方程</a></li>
  <li><a href="javascript:void(0)">复变函数</a></li>
  <li><a href="javascript:void(0)">抽象代数</a></li></ul>
 <ul id="sortable2" class="connectedSortable">计算机专业教材
  <li><a href="javascript:void(0)">数据结构</a></li>
  <li><a href="javascript:void(0)">数据库系统概论</a></li>
  <li><a href="javascript:void(0)">汇编语言</a></li>
  <li><a href="javascript:void(0)">操作系统设计与实现</a></li>
  <li><a href="javascript:void(0)">离散数学</a></li></ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#sortable1, #sortable2").sortable({connectWith: ".connectedSortable"}).disableSelection()用于向 connectWith 选项传递一个选择器,从而实现以拖曳的方式把一个列表中的元素排序到另一个列表中,反之亦然。sortable()方法是 jQuery UI 的可排序小部件(Sortable Widget)的方法,在任意的 DOM 元素上启用 sortable 功能,通过鼠标单击并拖曳元素到列表中的一个新的位置,其他条目就会自动调整。在默认情况下,sortable 的各个条目共享 draggable 属性。注意,在使用 sortable()方法实现拖动排序前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA133.html。

403 禁止或允许拖曳选项插入空列表中

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 sortable()方法中设置参数"connectWith: "ul", dropOnEmpty: false",从而实现禁止或允许以拖曳方式将选项插入空列表。当在 Google Chrome 浏览器中显示该页面时,如果使用鼠标拖动"计算机专业教材"列表中的"数据库系统概论"列表项到空列表"新的专业教材",如图 403-1 所示,然后松开鼠标,则"数据库系统概论"列表

项将不会被插入空列表“新的专业教材”中。如果使用鼠标拖动“数学基础教材”列表中的“常微分方程”列表项到空列表“新的专业教材”，如图 403-2 所示，然后松开鼠标，则“常微分方程”列表项将会被插入空列表“新的专业教材”中。当“新的专业教材”列表中有了列表项“常微分方程”之后，再使用鼠标拖动“计算机专业教材”列表中的“数据库系统概论”列表项到“新的专业教材”列表中，如图 403-3 所示，然后松开鼠标，则“数据库系统概论”列表项将会被插入“新的专业教材”列表中，如图 403-4 所示。有关此实例的主要代码如下。



图 403-1



图 403-2



图 403-3



图　403-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $( "ul.droptrue" ).sortable({                    //允许此列表项插入空列表
    connectWith: "ul" });
   $( "ul.dropfalse" ).sortable({                   //禁止此列表项插入空列表
    connectWith: "ul",  dropOnEmpty: false });
   $( "#sortable1, #sortable2, #sortable3" ).disableSelection();
  });
  </script>
  <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   .myContainer { width:620px; flow: left; }        /*设置盒子的基本样式*/
   /*设置无序列表的基本样式*/
   #sortable1, #sortable2, #sortable3 { list-style-type: none; margin: 0; padding: 0.5em 0; float:
left; margin-right: 10px; text-align: center; font-size: 24px; font-weight: bold; background-
color: darkgreen; color: yellow; border-radius:5px;}
   /*设置列表项的基本样式*/
   #sortable1 li, #sortable2 li, #sortable3 li { margin: 0 5px 1px 5px; padding: 5px; font-size:
1.2em; width: 175px;}
   /*设置列表项超链接的基本样式*/
   li a { -webkit-transition: 0.3s ease-out; background: lightgreen; color: black;
padding: 7px 15px 7px 10px; border-radius: 5px; width: 150px; display: block; text-decoration: none;
font-size: 14px; -webkit-box-shadow: 2px 2px 4px #888; }
 </style></head>
<body><div class="myContainer">
 <ul id="sortable1" class="droptrue">数学基础教材
  <li><a href="javascript:void(0)">数学分析</a></li>
  <li><a href="javascript:void(0)">高等代数</a></li>
  <li><a href="javascript:void(0)">常微分方程</a></li>
  <li><a href="javascript:void(0)">复变函数</a></li>
  <li><a href="javascript:void(0)">抽象代数</a></li></ul>
 <ul id="sortable2" class="dropfalse">计算机专业教材
  <li><a href="javascript:void(0)">数据结构</a></li>
  <li><a href="javascript:void(0)">数据库系统概论</a></li>
  <li><a href="javascript:void(0)">汇编语言</a></li>
  <li><a href="javascript:void(0)">操作系统设计与实现</a></li>
  <li><a href="javascript:void(0)">离散数学</a></li></ul>
 <ul id="sortable3" class="droptrue">新的专业教材</ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("ul.dropfalse").sortable({ connectWith: "ul", dropOnEmpty: false})表示禁止将$("ul.dropfalse")选项插入空列表。sortable()方法是 jQuery UI 的可排序小部件(Sortable Widget)的方法，在任意的 DOM 元素上启用 sortable 功能，通过鼠标单击并拖曳元素到列表中的一个新的位置，其他条目就会自动调整。注意，在使用 sortable()方法实现拖曳排序前必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA134.html。

404 以拖放风格实现在元素之间传递数据

此实例主要使用 dataTransfer 对象实现在两个元素之间通过拖放操作传递数据。当在 Google Chrome 浏览器中显示该页面时，在把文本拖放到文本框中之前效果如图 404-1 所示；在拖放操作完成之后，在文本框中将显示拖放的文本，原文本自动消失，效果如图 404-2 所示。有关此实例的主要代码如下。

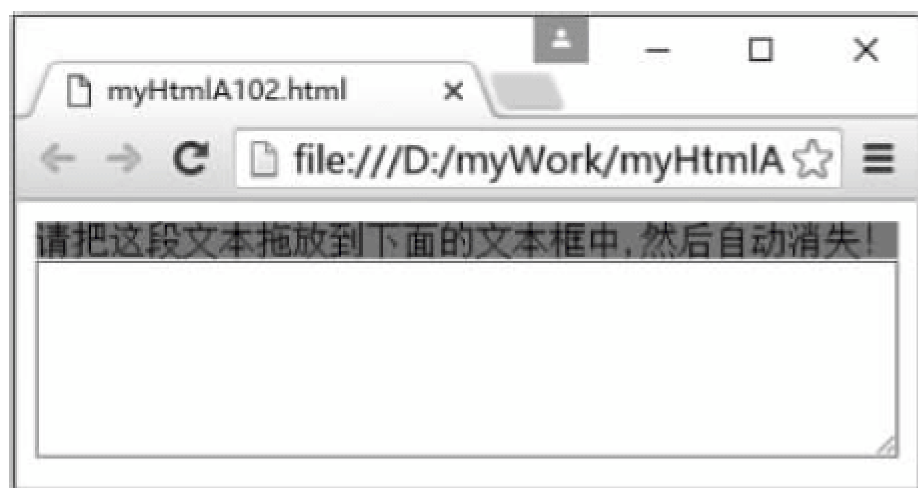


图 404-1



图 404-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function init(){
    var mySource = document.getElementById("mySource");
    var myTarget = document.getElementById("myTarget");
    mySource.addEventListener("dragstart", function(ev){//拖放开始
     var myData = ev.dataTransfer;                    //向 dataTransfer 对象追加数据
     myData.effectAllowed = 'all';
     myData.setData("text/plain", mySource.value);
    }, false);
    myTarget.addEventListener("dragend", function(ev){ //dragend: 拖放结束
      ev.preventDefault();                            //不执行默认处理(拒绝被拖放)
     }, false);
    myTarget.addEventListener("drop", function(ev){   //drop:被拖放(释放)
      var myData = ev.dataTransfer;                   //从 dataTransfer 对象那里取得数据
      myTarget.textContent += myData.getData("text/plain");
      ev.preventDefault();                            //不执行默认处理(拒绝被拖放)
      ev.stopPropagation();                           //停止事件传播
      $("#mySource").hide();                          //隐藏原文本
     }, false);  }
   //设置页面属性,不执行默认处理(拒绝被拖放)
    document.ondragover = function(e){e.preventDefault();};
    document.ondrop = function(e){e.preventDefault();}
 </script></head>
<body onload="init()"><div>
```

```
<output id="mySource"draggable="true"style="width:350px;height:20px; background: lightseagreen;">
请把这段文本拖放到下面的文本框中,然后自动消失!</output>
  <textarea id="myTarget" style="width:370px; height: 80px; border: 1px solid gray;"></textarea>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,dataTransfer 是拖曳数据传递对象,一般使用方式为 event.dataTransfer。dataTransfer.setData(format, data)方法用于设置传递的数据,其中参数 format 表示数据格式,参数 data 表示数据内容。dataTransfer.getData(format)方法用于获取传递的数据,其中参数 format 表示数据格式,返回值即是传递的数据。在实现拖放功能时通常需要处理相关的拖放事件,表 404-1 是对拖放相关事件的说明。

表 404-1 对拖放相关事件的说明

事件	产生事件的元素	说明
dragstart	被拖放的元素	开始拖放操作
drag	被拖放的元素	拖放过程中
dragenter	拖放过程中鼠标指针经过的元素	被拖放的元素开始进入本元素范围内
dragover	拖放过程中鼠标指针经过的元素	被拖放的元素正在本元素范围内移动
dragleave	拖放过程中鼠标指针经过的元素	被拖放的元素离开本元素的范围
drop	拖放的目标元素	有其他元素被拖放到了本元素中
dragend	拖放的对象元素	拖放操作结束

此实例的源文件名是 myHtmlA102.html。

405 创建两个滑块控件从两端改变数值范围

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 的 slider()方法创建包含两个滑块的范围控件,从而实现从两端改变数值范围的上限和下限值。当在 Google Chrome 浏览器中显示该页面时,使用鼠标向左拖动左边的滑块即可改变数值范围的下限值,如图 405-1 所示;使用鼠标向右拖动右边的滑块即可改变数值范围的上限值,如图 405-2 所示。有关此实例的主要代码如下。

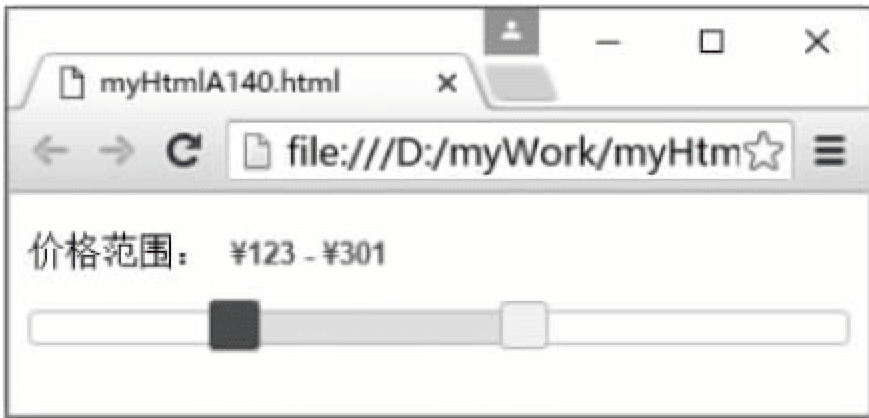


图 405-1

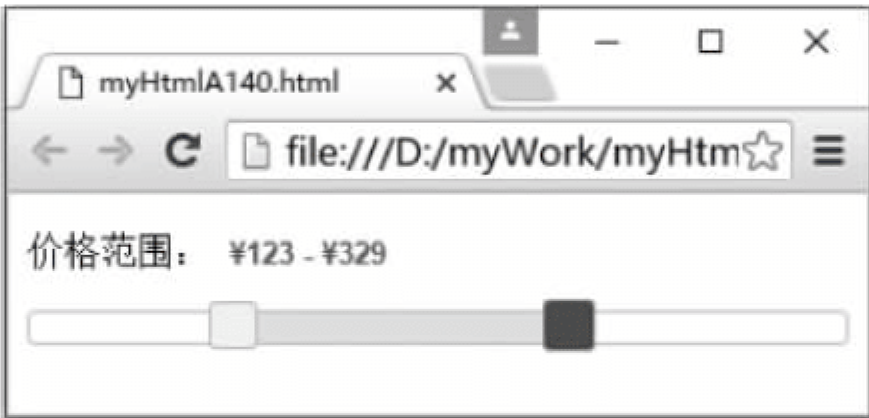


图 405-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myRange").slider({ range: true, min: 0, max: 500, values: [75, 300],
        slide: function (event, ui) {                          //显示范围两端的值
         $("#myPrice").val("¥" + ui.values[0] + " - ¥" + ui.values[1]);
    } });
    //显示预定义范围
```

```
    $("#myPrice").val("¥" + $("#myRange").slider("values", 0) + " - ¥" + $("#myRange").slider
("values", 1));
 });
 </script>
 <link rel = "stylesheet" href = "css/jquery-ui.min.css"><style type = "text/css">
  #myRange { margin: 3px auto;}                        /* 设置范围控件的基本样式 */
  #myPrice {border: 0;color: red;font-weight: bold;}   /* 设置提示文本的基本样式 */
 </style></head>
<body><p><label for = "myPrice">价格范围: </label>
   <input type = "text" id = "myPrice"></p><div id = "myRange"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myRange"). slider({range: true, min: 0,max: 500, values: [75, 300], slide: function (event, ui) { $("#myPrice").val("¥" + ui.values[0]+"-¥"+ui.values[1]);} })用于创建包含两个滑块的范围控件。slider()方法主要用于通过拖动单个滑块来改变数值，如果像实例这样设置其参数 range 为 true，则将出现两个滑块来改变数值范围的上限和下限值。slider()方法是滑块部件(Slider Widget)的方法，因此在使用 slider()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA140.html。

406 创建上限固定、下限可调节的范围滑块控件

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 的 slider()方法创建上限固定、下限可调节的范围滑块控件，从而实现拖动左端滑块改变数值范围的下限值。当在 Google Chrome 浏览器中显示该页面时，数值范围的上限是最大值，下限可调节，如图 406-1 所示；使用鼠标向左拖动滑块即可改变数值范围的下限值，如图 406-2 所示。有关此实例的主要代码如下。

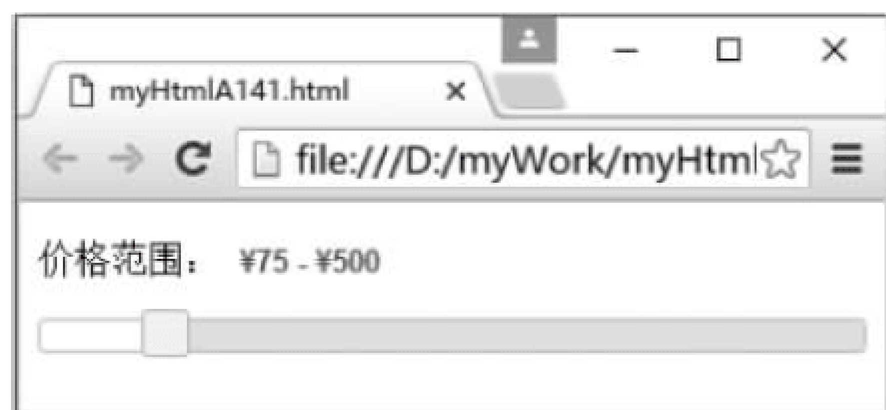


图 406-1

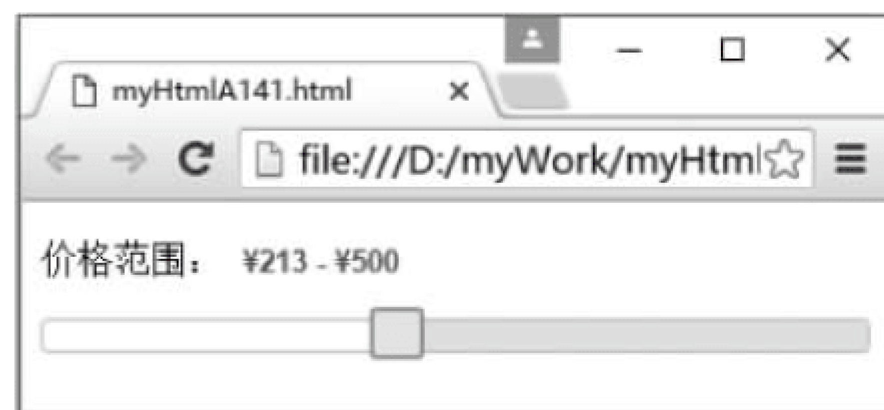


图 406-2

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script>
 <script src = "js/jquery-ui.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myRange").slider({ range: "max",//范围的最大值固定
      min: 0,//最小值
      max: 500,//最大值
      value: 75,//初始当前值
      slide: function( event, ui ) {                  //显示当前范围
        $( "#myPrice" ).val("¥" + ui.value + " - ¥500"); } });
  //显示预定义范围
   $( "#myPrice" ).val("¥" + $("#myRange" ).slider( "value") + " - ¥500" );
  });
```

```
</script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  #myRange { margin: 3px auto; }                            /* 设置范围控件的基本样式 */
  #myPrice { border: 0; color: red;font - weight: bold;}   /* 设置提示文本的基本样式 */
 </style></head>
<body><p><label for = "myPrice">价格范围: </label>
 <input type = "text" id = "myPrice"></p><div id = "myRange"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myRange").slider({range:"max",min:0,max:500,value:75,slide:function(event, ui) { $("#myPrice").val("￥"+ui.value+ "-￥500");}})用于创建上限固定、下限可调节的范围滑块控件，在这种情况下应该设置range参数为max。slider()方法主要用于实现通过拖动单个滑块来改变数值，该方法是滑块部件(Slider Widget)的方法，因此在使用slider()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。

此实例的源文件名是myHtmlA141.html。

407　创建下限固定、上限可调节的范围滑块控件

此实例主要使用jquery-ui.min.js和jquery-ui.min.css的slider()方法创建下限固定、上限可调节的范围滑块控件，从而实现拖动右端滑块改变数值范围的上限值。当在Google Chrome浏览器中显示该页面时，数值范围的下限是最小值，上限可调节，如图407-1所示；使用鼠标向右拖动滑块即可改变数值范围的上限值，如图407-2所示。有关此实例的主要代码如下。

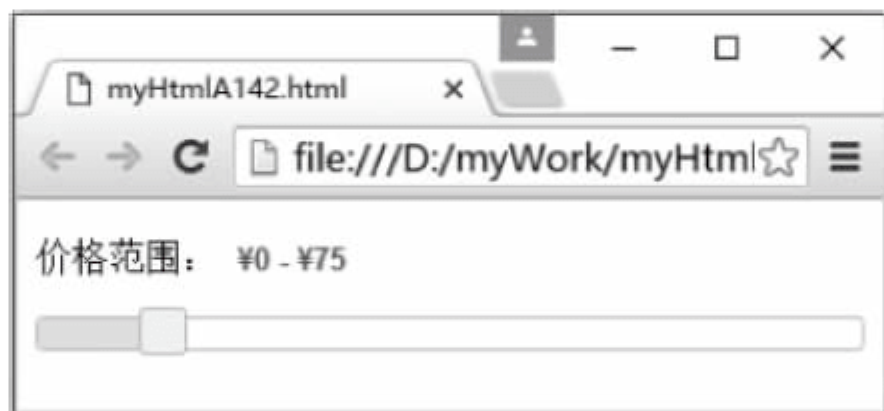


图　407-1

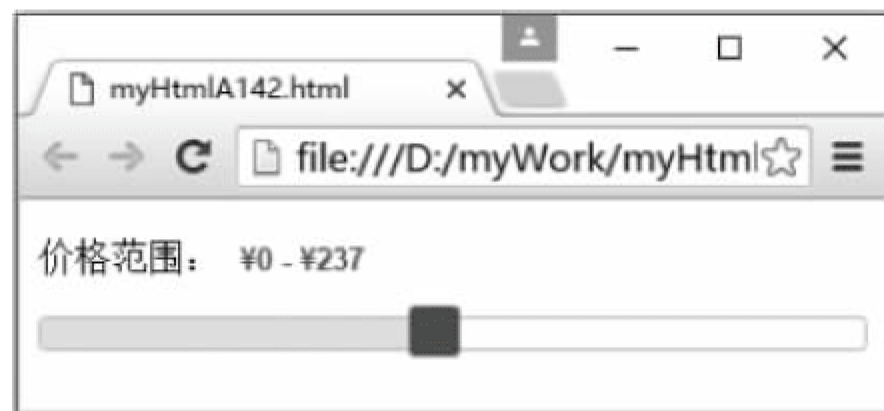


图　407-2

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myRange").slider({ range: "min",//范围的最小值固定
                          min: 0,//最小值
                          max: 500,//最大值
                          value: 75,//初始当前值
                          slide: function( event, ui ) {    //显示当前范围
                   $( "#myPrice" ).val("¥0 - ¥" + ui.value ); } });
  //显示预定义范围
   $( "#myPrice" ).val("¥0 - ¥" + $("#myRange" ).slider( "value"));
 });
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  #myRange{ margin: 3px auto; }                                 /* 设置范围控件的基本样式 */
```

```
   #myPrice{ border: 0; color: red; font-weight: bold;}     /*设置提示文本的基本样式*/
  </style></head>
<body><p><label for="myPrice">价格范围: </label>
   <input type="text" id="myPrice"></p><div id="myRange"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myRange").slider({range:"min",min:0,max:500,value:75,slide:function(event,ui){ $("#myPrice").val("￥0－￥"+ui.value);}})用于创建下限固定、上限可调节的范围滑块控件,在这种情况下应该设置 range 参数为 min。slider()方法主要用于实现通过拖动单个滑块来改变数值,该方法是滑块部件(Slider Widget)的方法,因此在使用 slider()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA142.html。

408　以拖动 range 滑块方式改变字体的大小

此实例主要设置<input>标签的 type 属性为 range,从而创建一个滑块,并通过拖动此滑块来实现改变字体的大小。当在浏览器中显示该页面时,如果使用鼠标向右拖动滑块,则演示文本“HTML5 炫酷实例集锦”逐渐变大;如果使用鼠标向左拖动滑块,则演示文本“HTML5 炫酷实例集锦”逐渐变小,如图 408-1 所示。有关此实例的主要代码如下。



图　408-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   function myFunc(){//获取滑块的当前值并转换成 int 类型
    var mySize=parseInt($("#mySize").prop("value"));
    //根据滑块值重置演示文本的字体大小
     $("p").css("font-size",mySize);
   }
   setInterval(myFunc,1);
  });
 </script></head>
<body><form style="width: 400px;">
 拖动滑块改变文字大小: <input type="range" id="mySize" min="12" max="60" value="12"/><br>
 <p style="font-size: 12px">HTML5 炫酷实例集锦</p></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,range 属性值是 HTML5 中<input>标签的 type 属性的新值,该属性值指定范围值的输入字段,在 HTML 页面中显示为滑块。一旦将<input>标签的 type 属性设置为 range,则应该参考表 408-1 设置<input>标签的相关属性,从而指定

该滑块的有效范围和步长值。

表 408-1　设置<input>标签的相关属性

属　性	值	描　述
max	数字	规定允许的最大值
min	数字	规定允许的最小值
step	数字	规定合法的数字间隔(如果 step="3",则合法的数是-3、0、3、6 等)
value	数字	规定默认值

此实例的源文件名是 myHtmlA010.html。

409　创建图形和文字结合的进度条显示进程

此实例主要通过使用<progress>标签实现使用图形和文字结合的进度条动态显示当前的进程。当在浏览器中显示该页面时,单击"启动进度条"按钮,则进度条将动态改变当前的状态值,并用文字显示完成百分比,如图 409-1 所示。有关此实例的主要代码如下。

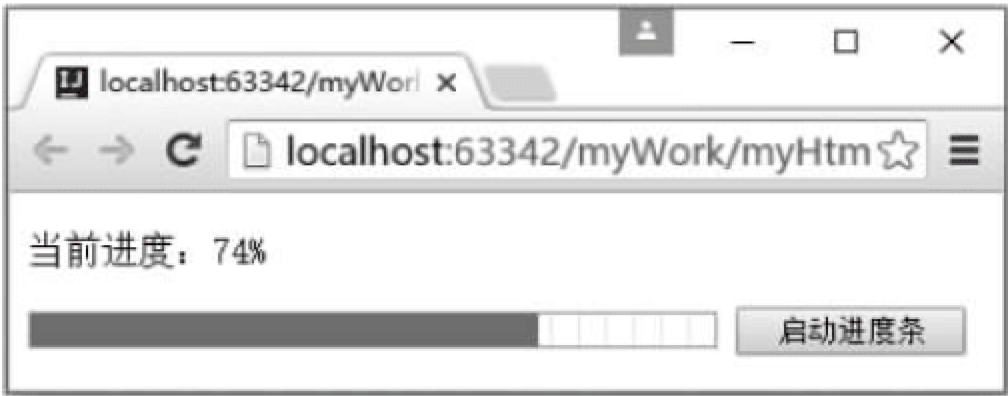


图　409-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   var myValue=0; var myTimer;
   function myFunc() {
    myValue++; $("#myBar").val(myValue);
    if (myValue>=$("#myBar").prop("max")) {
     clearInterval(myTimer);
     $("#myInfo").html("下载完成")  ;
    } else {
     $("#myInfo").html("当前进度: "+myValue+"%") ;
  } }
  $("#myBtn").click(function() {//启动进度条
   myValue=0; myTimer=setInterval(myFunc, 100);
  });});
 </script></head>
<body><p id="myInfo"> 当前进度: </p>
<progress value="0" max="100" id="myBar" style="width:300px"></progress>
<input type="button" value="启动进度条" id="myBtn" style="width:100px"/>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<progress>标签定义运行中的进度(进程),该标签是 HTML5 的新标签。<progress>标签(元素)最常用的属性是 max 和 value,max 属性定义最大值,value 属性定义进程的当前值。

此实例的源文件名是 myHtmlA004.html。

410 使用 meter 模拟 progress 的进度显示

此实例主要通过使用 meter 元素模拟 progress 元素的功能来动态显示进度。当在浏览器中显示该页面时，单击“启动进度条”按钮，则进度条将动态改变当前的状态值，并用文字显示完成百分比，如图 410-1 所示。有关此实例的主要代码如下。

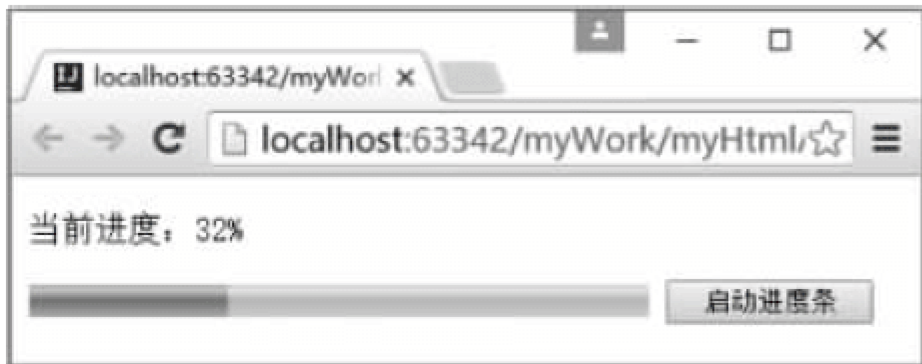


图 410-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    var myValue = 0; var myTimer;
    function myFunc() {
     myValue++;
     $("#myBar").val(myValue);
     if (myValue >= $("#myBar").prop("max")) {
      clearInterval(myTimer);
      $("#myInfo").html("下载完成")  ;
     } else {
      $("#myInfo").html("当前进度：" + myValue + "%");
  } }
   $("#myBtn").click(function() {//启动进度条
    myValue = 0; myTimer = setInterval(myFunc, 100);
 });});
 </script></head>
<body><p id = "myInfo">当前进度：</p>
<meter min = "0" max = "100" value = "0" id = "myBar"  style = "width:300px" ></meter>
<input type = "button" value = "启动进度条" id = "myBtn" style = "width:100px">
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<meter>标签显示运行中的进度(进程)，该标签是 HTML5 的新标签。<meter>标签仅用于已知最大和最小值的度量。<meter>标签常用的属性说明如表 410-1 所示。

表 410-1 <meter>标签常用的属性及说明

属　性	值	描　述
high	number	定义度量的值位于哪个点，被界定为高的值
low	number	定义度量的值位于哪个点，被界定为低的值
max	number	定义最大值，默认值是 1
min	number	定义最小值，默认值是 0
optimum	number	定义什么样的度量值是最佳的值。如果该值高于 high 属性的值，则意味着值越高越好；如果该值低于 low 属性的值，则意味着值越低越好
value	number	定义度量的值

此实例的源文件名是 myHtmlA026.html。

411　在进度条上显示进度完成的百分比值

此实例主要在 jquery-ui. min. js 和 jquery-ui. min. css 的 progressbar ()方法中传递参数，从而实现通过定时器函数 setTimeout()在进度条上同步显示当前进度的百分比值。当在 Google Chrome 浏览器中显示该页面时，进度条工作时显示的百分比值如图 411-1 所示，进度完成之后的进度条显示的提示信息如图 411-2 所示。有关此实例的主要代码如下。

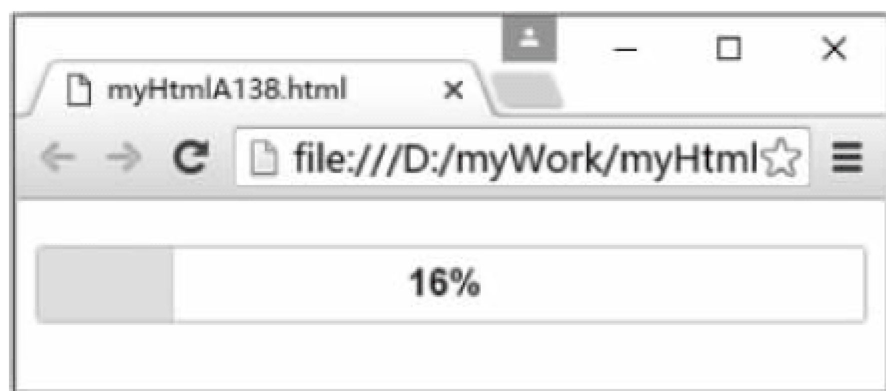


图　411-1

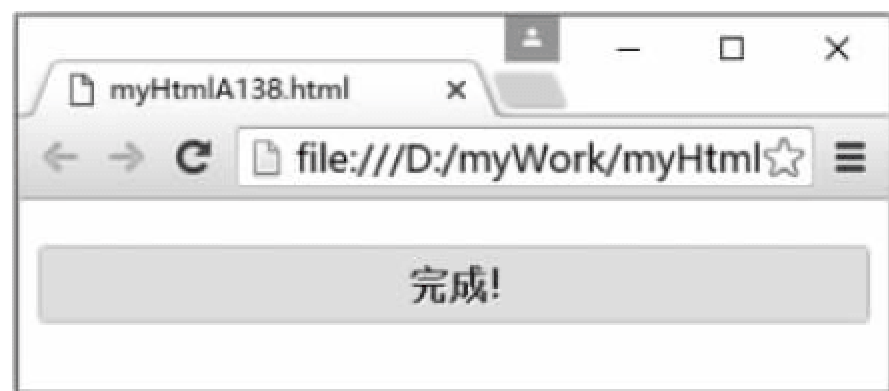


图　411-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   var myBar = $( "#myBar" ), myLabel = $( ".myLabel" );
    myBar.progressbar({                              //创建图文结合的进度条
     value: false,
     change: function() {myLabel.text( myBar.progressbar( "value" ) + "%" ); },
     complete: function() { myLabel.text( "完成!" );} //进度结束的提示文本
   });
   function progress() {
    var val = myBar.progressbar( "value" ) || 0;
    myBar.progressbar( "value", val + 1 );         //设置当前进度值
     if ( val < 99 ) {                             //使用定时器每隔 100 毫秒更新一次进度
      setTimeout(progress, 100 );
   } }
   setTimeout( progress, 1 );                      //使用定时器每隔 1 毫秒更新一次进度
  });
  </script>
  <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   /* 设置进度提示文本的基本样式 */
   .myLabel { position: absolute;left: 45%; top: 28px; font-weight: bold; text-shadow: 1px 1px 0 #FFF; }
    #myBar{ margin:20px auto; }
  </style></head>
<body><div id="myBar"><div class="myLabel">加载...</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，setTimeout(progress, 1)表示每毫秒调用一次 progress()方法。在 JavaScript 中，setTimeout()函数用于在指定的毫秒数后调用函数(方法)或计算表达式，该函数的语法格式如下。

```
setTimeout(code,millisec)
```

其中，参数 code 表示在调用函数后要执行的 JavaScript 代码串；参数 millisec 表示在执行代码前需等待的毫秒数。注意，setTimeout()这个定时器只执行一次 code，如果要多次调用，应使用

setInterval()或者让 code 自身再次调用 setTimeout()。

myBar. progressbar()用于设置进度条工作时的相关参数。progressbar()方法主要用于显示一个确定的或不确定的进程状态。该方法是进度条部件(Progressbar Widget)的方法,因此在使用 progressbar()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA138. html。

412 使用随机色设置不确定进度条的背景颜色

此实例主要在 jquery-ui. min. js 和 jquery-ui. min. css 的 progressbar()方法中传递参数 false 创建不确定进度条,然后使用 Math. random()方法产生随机颜色值,从而实现使用随机色设置不确定进度条的背景颜色。当在 Google Chrome 浏览器中显示该页面时,不确定进度条的背景颜色如图 412-1 所示;按 F5 键刷新页面,则使用另一种随机颜色设置不确定进度条的背景颜色,如图 412-2 所示。有关此实例的主要代码如下。

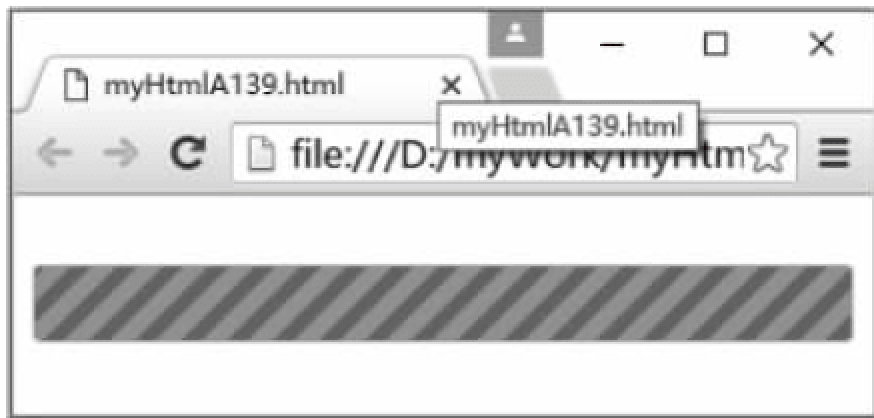


图 412-1

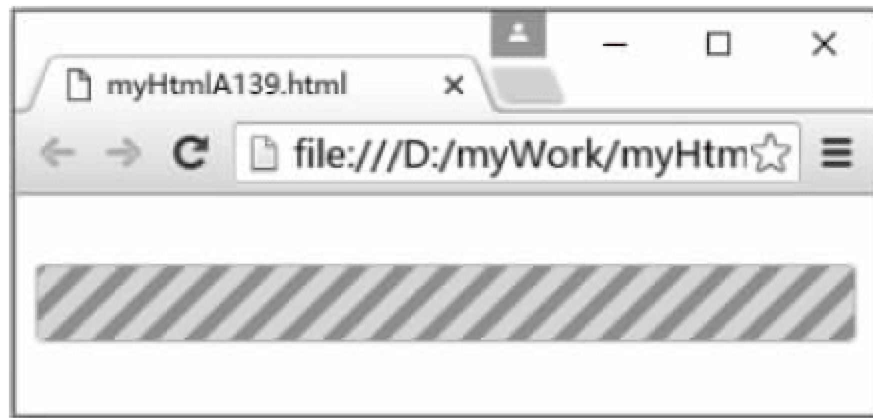


图 412-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBar").progressbar({ value: false });
    var myValue=$("#myBar").find(".ui-progressbar-value");
    myValue.css({"background": '#'+Math.floor(Math.random()*16777215).toString(16)});   //使用随机
颜色设置不确定进度条的背景颜色
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   #myBar {margin: 30px auto;}   /* 设置进度条的基本样式 */
 </style></head>
<body><div id="myBar"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,Math. random()方法用于返回 0～1 的一个随机数;Math. floor()方法用于进行向下取整计算,它返回的是小于或等于函数参数,并且与之最接近的整数;$("#myBar"). progressbar({value: false})用于创建一个不确定的进度条。progressbar()方法主要用于显示一个确定的或不确定的进程状态,该方法是进度条部件(Progressbar Widget)的方法,因此在使用 progressbar()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA139. html。

413　通过设置按钮的阴影属性创建渐变色按钮

此实例主要通过在 CSS 样式中设置按钮的 box-shadow 属性来创建渐变色按钮。当在 Google Chrome 浏览器中显示该页面时，创建的两个渐变色按钮如图 413-1 所示。有关此实例的主要代码如下。



图　413-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  input {color: yellow;height: 34px;width: 200px;border-radius: 34px;
        box-shadow: inset 0 1px #FFF, inset 0 12px rgba(255,255,255,0.15), inset 0 4px 10px #CEF,
inset 0 22px 5px #0773B4, inset 0 -5px 10px #0DF;}
 </style></head>
<body><input type="button" value="HTML5 炫酷应用实例集锦"/>
      <input type="button" value="CSS3 炫酷应用实例集锦"/></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：inset 0 1px #FFF，inset 0 12px rgba(255,255,255,0.15)，inset 0 4px 10px #CEF，inset 0 22px 5px #0773B4，inset 0 -5px 10px #0DF 用于创建渐变色按钮，为了使其达到逐级渐变颜色的效果，使用了多个内置阴影。box-shadow 的语法格式如下。

```
box-shadow: inset  x-offset y-offset blur-radius spread-radius color
```

在默认情况下，box-shadow 是外置阴影，因此可以省略 inset，当需要设置内置阴影时则应明确指定 inset。

此实例的源文件名是 myHtmlB049.html。

414　通过设置按钮的阴影属性创建中心扩散按钮

此实例主要通过在 CSS 样式中设置按钮的 box-shadow 属性来创建颜色向中心扩散淡化的按钮。当在 Google Chrome 浏览器中显示该页面时，创建的两个颜色向中心扩散的按钮如图 414-1 所示。有关此实例的主要代码如下。



图　414-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  input { height: 50px; width: 200px; border-radius: 30px;
          box-shadow: inset 0px 0px 30px 10px rgba(0,51,53,0.85);}
  body{background-color: white;}
 </style></head>
<body><input type="button" value="HTML5 炫酷应用实例集锦"/>
      <input type="button" value="CSS3 炫酷应用实例集锦"/></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：inset 0px 0px 30px 10px rgba(0,51,53,0.85)用于创建颜色向中心扩散的按钮，为了达到颜色向中心扩散的效果，使用了内置阴影 inset。box-shadow 的语法格式如下。

```
box-shadow: inset  x-offset y-offset blur-radius spread-radius color
```

在默认情况下，box-shadow 是外置阴影，因此可以省略 inset，当需要设置内置阴影时则应明确指定 inset。在此实例中两个 0px 分别代表 x-offset 和 y-offset，30px 表示 blur-radius，10px 表示 spread-radius，rgba(0,51,53,0.85)表示 color。

此实例的源文件名是 myHtmlB051.html。

415　使用盒子阴影创建金属质感的立体按钮

此实例主要通过使用盒子阴影和径向渐变创建带金属质感的立体按钮。当在 Google Chrome 浏览器中显示该页面时，单击“赞成”按钮时效果如图 415-1 所示；单击“反对”按钮时效果如图 415-2 所示；单击“弃权”按钮时效果如图 415-3 所示。有关此实例的主要代码如下。



图　415-1



图　415-2



图　415-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
    /*设置按钮正常状态时的阴影*/
    .myButton { text-decoration: none; font: 24px/1em 'Droid Sans', sans-serif;
               font-weight: bold; text-shadow: rgba(255, 255, 255, 0.5) 0 1px 0;padding:
               0.5em  0.6em 0.4em 0.6em; margin: 0.5em; display: inline-block; position: relative; -
               webkit-border-radius: 8px; border-top: 1px solid rgba(255, 255, 255, 0.8); border-
               bottom: 1px solid rgba(0, 0, 0, 0.1);
      /*通过径向渐变生成按钮表面的金属质感的发光效果*/
      background-image: -webkit-gradient(radial, 50% 0, 100, 50% 0, 0, from(rgba(255, 255, 255,
      0)), to(rgba(255, 255, 255, 0.7)));
      color: hsl(0, 0%, 40%)!important; background-color: hsl(0, 0%, 75%);
      /*生成下边框线的立体阴影*/
      -webkit-box-shadow: inset rgba(255, 254, 255, 0.6) 0 0.3em 0.3em, inset rgba(0, 0, 0, 0.15) 0
      -0.1em 0.3em, hsl(0, 0%, 60%) 0 0.1em 3px, hsl(0, 0%, 45%) 0 0.3em 1px, rgba(0, 0, 0, 0.2) 0
      0.5em 5px;}
    /*设置按钮被按下时的阴影*/
    .myButton:active { background-image: -webkit-gradient(radial, 50% 0, 100, 50% 0, 0, from(rgba
                     (255, 255, 255, 0)), to(rgba(255, 255, 255, 0))); -webkit-box-shadow: inset
                     rgba(255, 255, 255, 0.6) 0 0.3em 0.3em, inset rgba(0, 0, 0, 0.2) 0 -0.1em 0.3em,
                     rgba(0, 0, 0, 0.4) 0 0.1em 1px, rgba(0, 0, 0.2) 0 0.2em 6px; -webkit-
                     transform: translateY(0.2em);}
  div{ margin: 15px; }
 </style></head>
<body><div><a href="#" class="myButton" onclick="alert('赞成')">赞成</a><a href="#" class="
myButton" onclick="alert('反对')">反对</a><a href="#" class="myButton" onclick="alert('弃权')">
弃权</a></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-color：hsl(0，0%，75%)表示以HSL记法设置背景颜色，此处主要用于产生发光的银色金属质感。在CSS3中，HSL记法的语法格式如下。

```
HSL(H,S,L)
```

其中，H表示Hue(色调)，0(或360)表示红色、120表示绿色、240表示蓝色，用户也可取其他数值来指定颜色，取值范围为0～360；S表示Saturation(饱和度)，取值范围为0.0%～100.0%；L表示Lightness(亮度)，取值范围为0.0%～100.0%。

-webkit-box-shadow：inset rgba(255，254，255，0.6) 0 0.3em 0.3em，inset rgba(0，0，0，0.15) 0 －0.1em 0.3em，hsl(0，0%，60%) 0 0.1em 3px，hsl(0，0%，45%) 0 0.3em 1px，rgba(0，0，0，0.2) 0 0.5em 5px表示以多级阴影的形式生成按钮下边框线的立体效果。在CSS3中，box-shadow属性用于向盒子添加一个或多个阴影，该属性是由逗号分隔的阴影列表，每个阴影由2～4个长度值、可选的颜色值以及可选的inset关键字来规定，省略长度的值是0。

此实例的源文件名是myHtmlB312.html。

416 将默认的3D风格按钮重置为扁平化按钮

此实例主要取消默认按钮的边框样式并使用内置阴影重建边框，从而实现将默认的3D风格按钮重置为扁平化按钮。当在Google Chrome浏览器中显示该页面时，上面的按钮是自定义的扁平化按钮，下面的按钮是默认的3D风格按钮，如图416-1所示。有关此实例的主要代码如下。

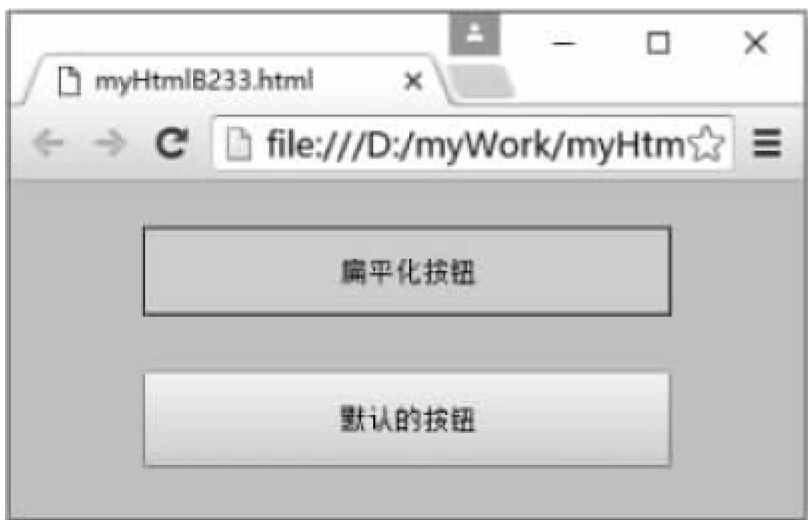


图 416-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  body{ background-color: lightgray; }
  .myButton { /* 通过内置阴影定制边框线 */
              box-shadow: inset 0 0 0 1px black; border-style: none; }
  button{ margin: 1em; padding: 1em 2em;text-align: center;  width:250px; }
 </style></head>
<body><center><div><button class="myButton">扁平化按钮</button><br>
     <button>默认的按钮</button></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：inset 0 0 0 1px black 用于创建宽度为 1px 的黑色内置阴影；border-style：none 表示无边框。在 CSS 中，border-style 属性用于设置元素所有边框的样式，或者单独为各边设置边框样式，该属性支持的属性值的意义如表 416-1 所示。

表 416-1 border-style 属性支持的属性值

属 性 值	意 义
none	定义无边框
hidden	与 none 相同，不过应用于表时除外，对于表，hidden 用于解决边框冲突
dotted	定义点状边框，在大多数浏览器中呈现为实线
dashed	定义虚线，在大多数浏览器中呈现为实线
solid	定义实线
double	定义双线，双线的宽度等于 border-width 的值
groove	定义 3D 凹槽边框，其效果取决于 border-color 的值
ridge	定义 3D 垄状边框，其效果取决于 border-color 的值
inset	定义 3D inset 边框，其效果取决于 border-color 的值
outset	定义 3D outset 边框，其效果取决于 border-color 的值
inherit	规定应该从父元素继承边框样式

此实例的源文件名是 myHtmlB233.html。

417 使用 radio 单选按钮隐藏或者显示内容

此实例主要通过使用 radio 单选按钮（input 元素）实现隐藏或显示图片等内容。当在 Google Chrome 浏览器中显示该页面时，单击“显示图片”，则图文混合效果如图 417-1 所示；单击“隐藏图片”，则图片隐藏之后的效果如图 417-2 所示。有关此实例的主要代码如下。



图　417-1

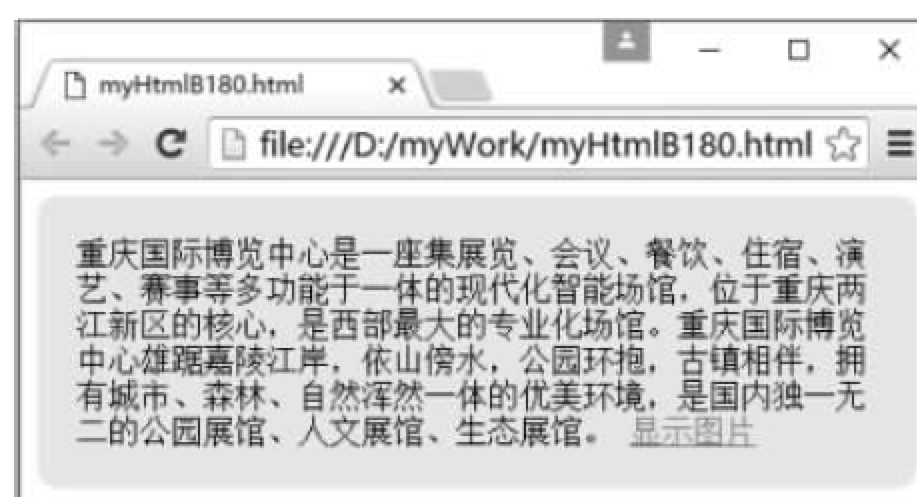


图　417-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .myBox { width: 445px;}
  .myContent { background-color: #EEE; padding: 20px; border-radius: 10px;}
  .myCheckbox, .myMore, .myHide { position: absolute; left: -9999px;}
  .myCheckbox:checked ~ .myLabel .myShow { position: absolute;left: -9999px;}
  .myLabel {cursor: pointer; color: lightseagreen;text-decoration: underline;}
  .myLabel:hover {text-decoration: none;}
  .myCheckbox:checked ~ .myMore, .myCheckbox:checked ~ .myLabel .myHide {
            position: static;}
  img { width: 400px; height: 250px;border-radius: 10px;
        margin-top: 10px; margin-bottom: 5px;}
 </style></head>
<body><div class="myBox"><div class="myContent">重庆国际博览中心是一座集展览、会议、餐饮、住宿、演艺、赛事等多功能于一体的现代化智能场馆,位于重庆两江新区的核心,是西部最大的专业化场馆。重庆国际博览中心雄踞嘉陵江岸,依山傍水,公园环抱,古镇相伴,拥有城市、森林、自然浑然一体的优美环境,是国内独一无二的公园展馆、人文展馆、生态展馆。
  <input type="checkbox" id="myToggle" class="myCheckbox"/>
  <span class="myMore"><img src="img/B180.jpg"/></span>
  <label class="myLabel" for="myToggle">
   <span class="myHide">隐藏图片</span><span class="myShow">显示图片</span>
  </label></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，.myCheckbox，.myMore，.myHide{position：absolute；left：-9999px；}用于在未选中单选按钮myToggle时将“隐藏图片”文本和图片放置在屏幕之外，即用户不可见“隐藏图片”文本和图片；.myCheckbox：checked ～.myLabel.myShow{position：absolute；left：-9999px；}用于在选中单选按钮myToggle时将“显示图片”文本放置在屏幕之外，即用户不可见“显示图片”文本。

此实例的源文件名是myHtmlB180.html。

418 使用 appearance 属性将超链接显示为按钮

此实例主要通过设置超链接的 appearance 属性实现超链接以按钮风格进行显示。当在 Google Chrome 浏览器中显示该页面时，6 个按钮本质上就是超链接，单击每个按钮则会跳转到相应公司的官方网站，如图 418-1 所示。有关此实例的主要代码如下。



图 418-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a{ margin: 5px; padding:5px; text-decoration: none ;width: 350px;
     text-align: center;font-size: 14px; -webkit-appearance:button;}
  p{ width: 350px; text-align: center;}
 </style></head>
<body><p>2016 年中国 500 强企业前 6 名名单<p>
<a href="http://www.sgcc.com.cn">国家电网公司</a>
<a href="http://www.cnpc.com.cn">中国石油天然气集团公司</a>
<a href="http://www.sinopecgroup.com">中国石油化工集团公司</a>
<a href="http://www.icbc.com.cn">中国工商银行股份有限公司</a>
<a href="http://www.ccb.com">中国建设银行股份有限公司</a>
<a href="http://www.cscec.com.cn">中国建筑股份有限公司</a></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，appearance 属性主要用于重置 HTML5 表单元素浏览器内置的一些 UI，webkit 支持的 appearance 属性值有 checkbox、scrollbarbutton-up、scrollbarbutton-down、scrollbarbutton-left、slider-horizontal、searchfield-decoration、scrollbarbutton-right、scrollbartrack-horizontal、button、scrollbartrack-vertical、scrollbarthumb-horizontal、sliderthumb-horizontal、radio、scrollbarthumb-vertical、scrollbargripper-horizontal、scrollbargripper-vertical、square-button、textfield、slider-vertical、sliderthumb-vertical、listitem、menulist、searchfield-results-decoration、searchfield-results-button、menulist-textfield、searchfield-cancel-button、menulist-menulist-text、searchfield、textarea、button-bevel、listbox、push-button、caret。如果需要去掉 HTML5 表单元素的默认样式，可以直接设置该元素的-webkit-appearance 属性为 none。

此实例的源文件名是 myHtmlB132.html。

419　在日期选择器中实现年份和周数的选择

此实例主要设置<input>标签的 type 属性为 week，从而实现允许用户在日期选择器中选择年份和周数。当在浏览器中显示该页面时，单击"实习周数："文本框右端的下拉箭头，将滑出日期选择器，在日期选择器中选择年份和周数，例如"2016 年第 51 周"，如图 419-1 所示，则选择结果将显示在"实习周数："文本框中；单击"确认登记信息"按钮，将在弹出的消息框中以纯文本的形式显示该年份和周数，如图 419-2 所示。当然，用户也可以直接在"实习周数："文本框中修改年份和周数。有关此实例的主要代码如下。

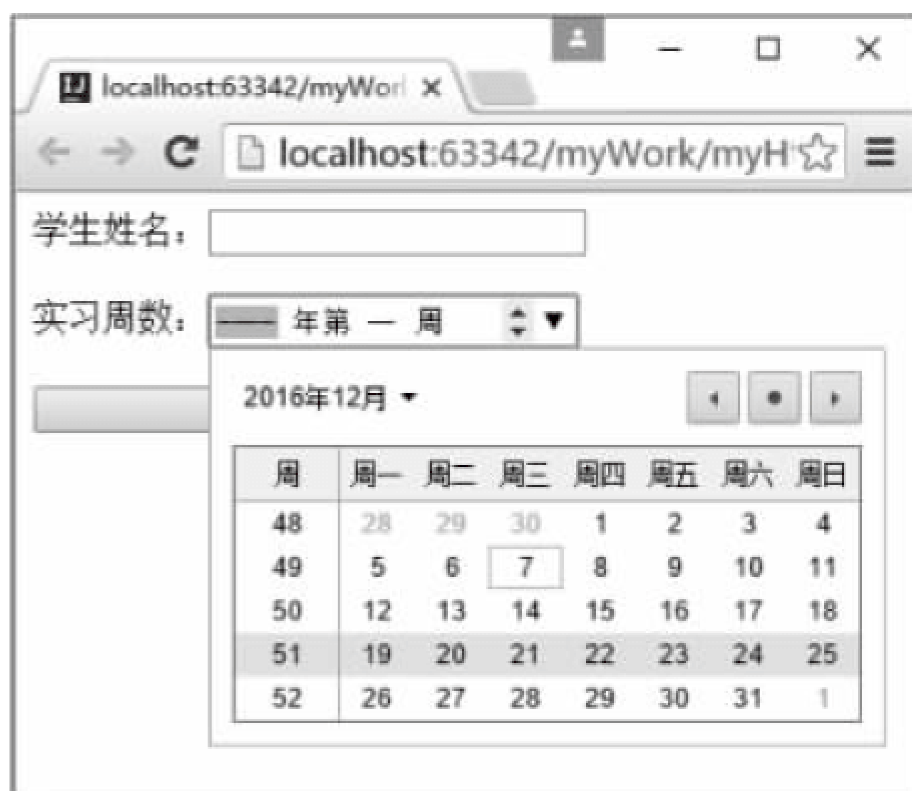


图　419-1

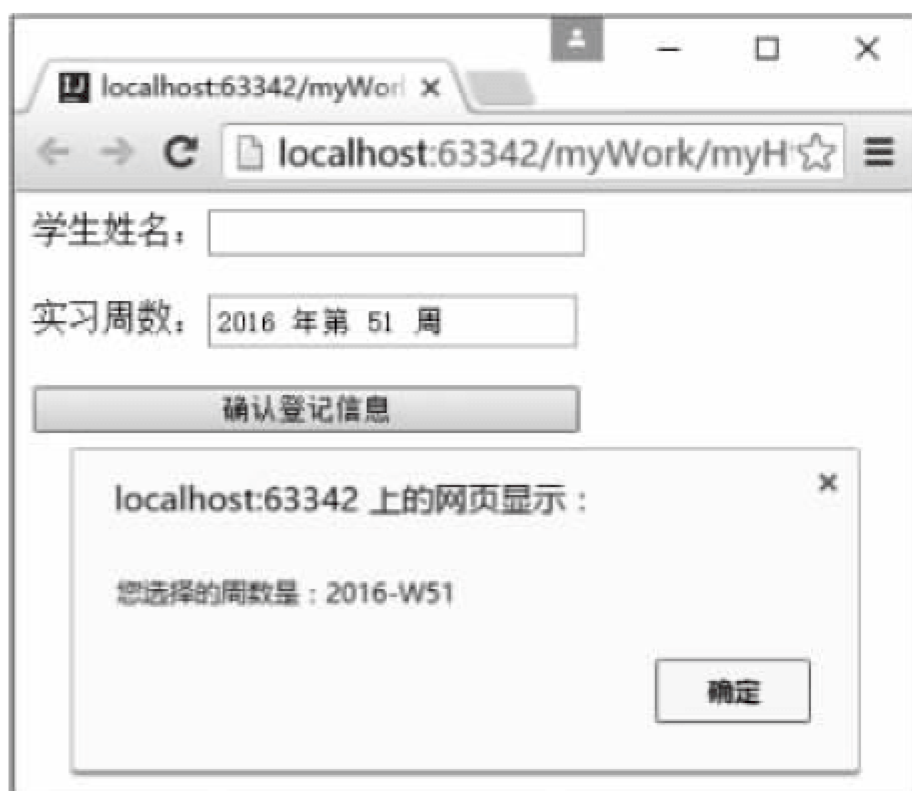


图　419-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtn").click(function() {//确认登记信息
     alert("您选择的周数是："+$("#myWeek").prop("value"));
   }) });
 </script></head>
<body><form style="width: 400px;">
   学生姓名：<input type="text" id="myName"/><br><br>
   实习周数：<input type="week" id="myWeek" /><br><br>
   <button type="button" id="myBtn" style="width: 250px">确认登记信息</button>
 </form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，week 属性值是 HTML5 中<input>标签的 type 属性的新值，该属性值能够使用户在日期选择器中选择年份和周数，可以通过<input>标签的 value 属性来获取年份和周数信息。

此实例的源文件名是 myHtmlA011.html。

420　在日期选择器中实现年份和月数的选择

此实例主要设置<input>标签的 type 属性为 month，从而实现允许用户在日期选择器中选择年份和月数。当在浏览器中显示该页面时，在"实习月数："文本框中将显示预置的年份和月数，例如

“2016 年 12 月”，用户可以直接在此编辑年份和月数，也可以单击右端的下拉箭头，滑出日期选择器，在日期选择器中选择年份和月数，如图 420-1 所示，选择结果将显示在“实习月数:”文本框中；单击“确认登记信息”按钮，将在弹出的消息框中以纯文本的形式显示该年份和月数，如图 420-2 所示。有关此实例的主要代码如下。

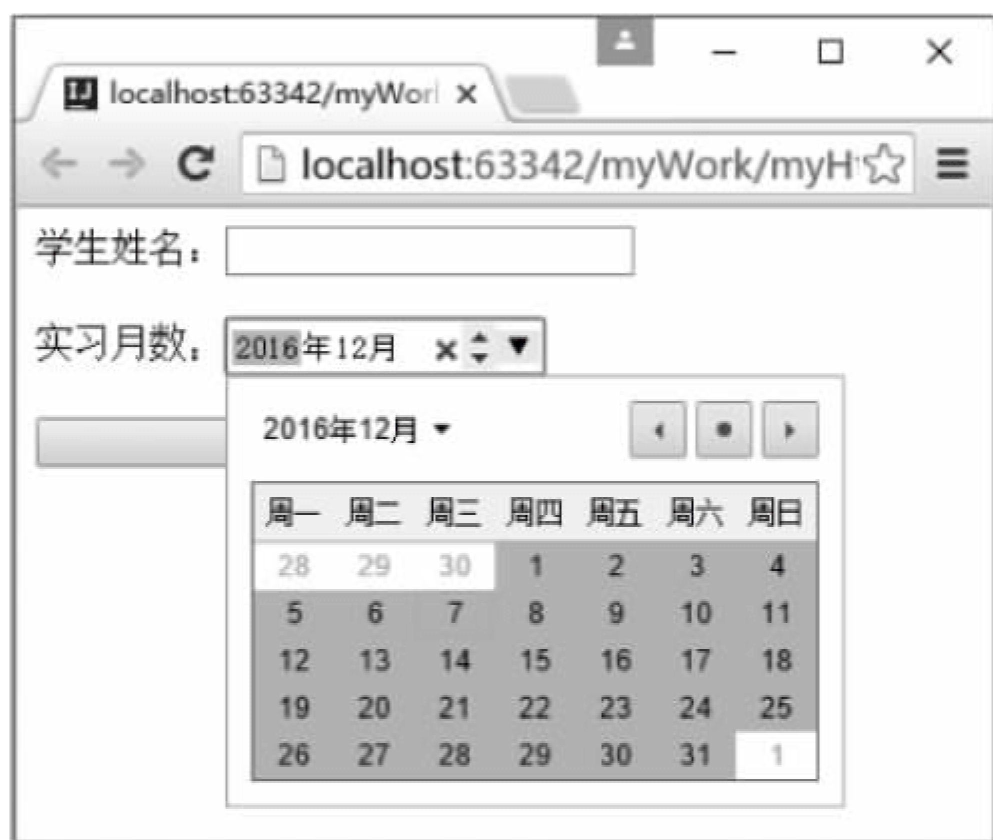


图 420-1

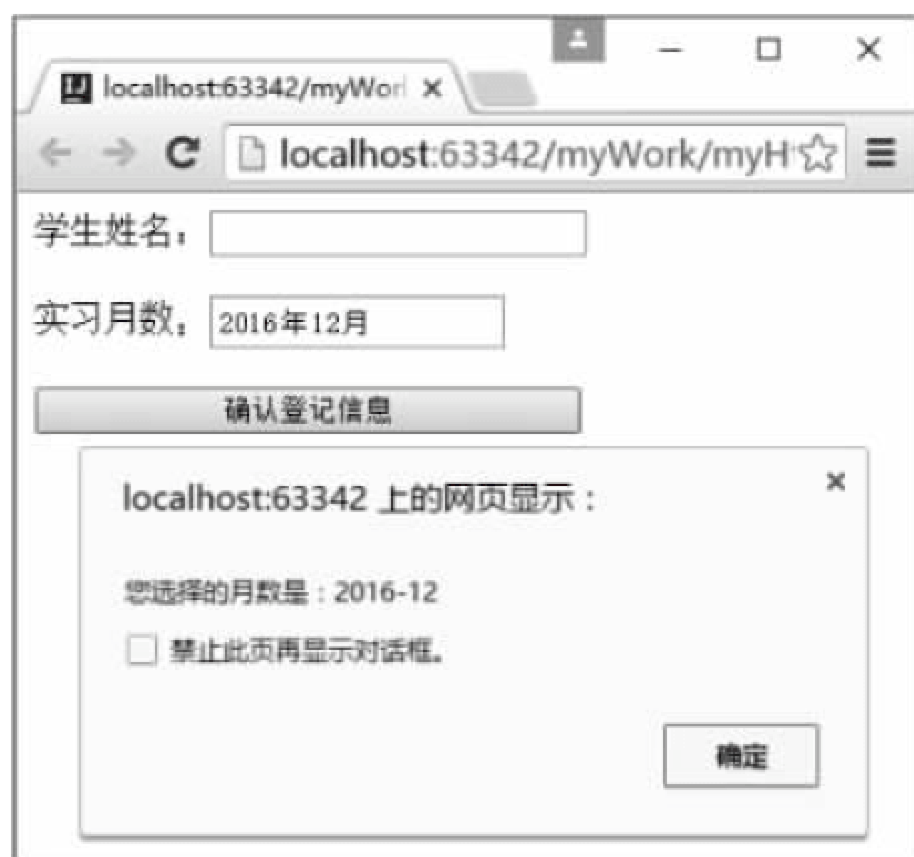


图 420-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {//确认登记信息
    alert("您选择的月数是: "+$("#myMonth").prop("value"));
  }) });
 </script></head>
<body><form style="width: 400px;">
   学生姓名: <input type="text" id="myName"/><br><br>
   实习月数: <input type="month" id="myMonth" value="2016-12"/><br><br>
   <button type="button" id="myBtn" style="width: 250px">确认登记信息</button>
 </form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，month 属性值是 HTML5 中<input>标签的 type 属性的新值，该属性值能够使用户在日期选择器中选择年份和月数，可以通过<input>标签的 value 属性来获取年份和月数信息。

此实例的源文件名是 myHtmlA012.html。

421 在日期选择器中实现年、月、日信息的选择

此实例主要设置<input>标签的 type 属性为 date，从而实现允许用户在日期选择器中选择年、月、日信息。当在浏览器中显示该页面时，在“出生日期:”文本框中将显示预置的日期信息，例如“1987/08/23”，用户可以直接在此修改日期信息，也可以单击右端的下拉箭头，滑出日期选择器，在日期选择器中选择日期信息，如图 421-1 所示，选择结果将显示在“出生日期:”文本框中；单击“确认登记信息”按钮，将在弹出的消息框中以纯文本的形式显示该日期信息，如图 421-2 所示。有关此实例的主要代码如下。



图　421-1

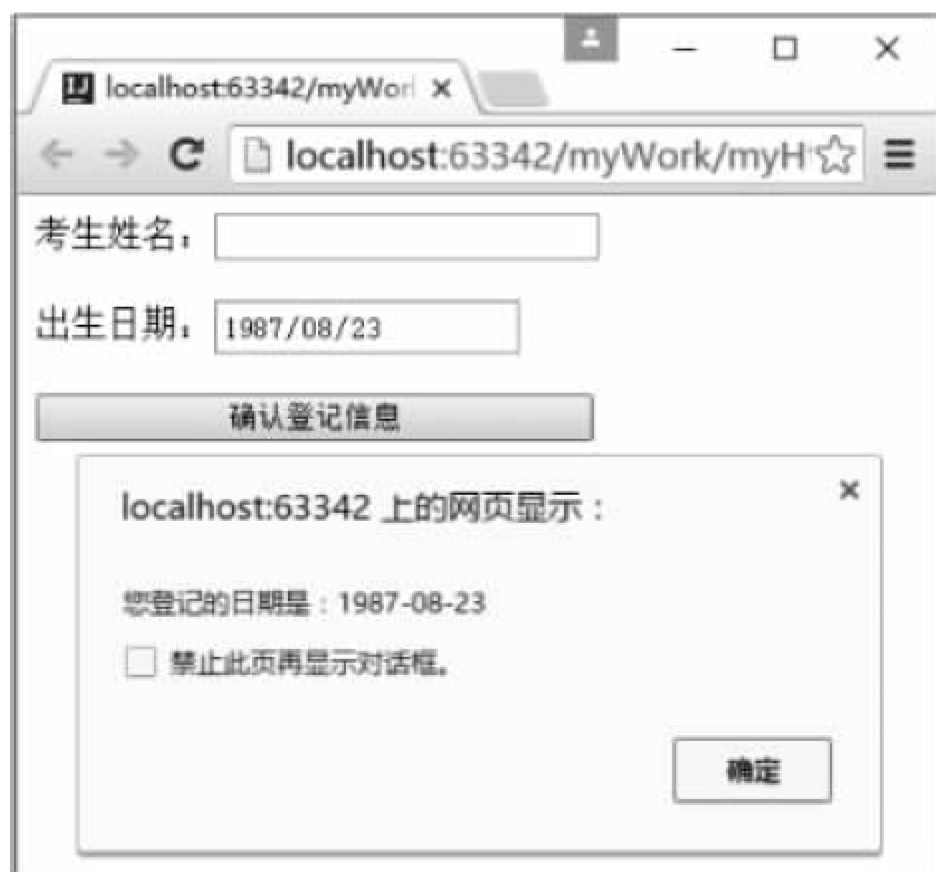


图　421-2

```
<!doctype html><html><head><meta charset = UTF - 8 >
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $ (document).ready(function() {
   $ ("#myBtn").click(function() {//确认登记信息
    alert("您登记的日期是: " + $ ("#myDate").prop("value"));
  }) });
</script></head>
<body><form style = "width: 400px;">
   考生姓名: <input type = "text" id = "myName"/><br><br>
   出生日期: <input type = "date" id = "myDate" value = "1987 - 08 - 23"/><br><br>
   <button type = "button" id = "myBtn" style = "width: 250px">确认登记信息</button>
</form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，date 属性值是 HTML5 中< input >标签的 type 属性的新值，该属性值能够使用户在日期选择器中选择年、月、日等日期信息，可以通过< input >标签的 value 属性来获取该日期信息。

此实例的源文件名是 myHtmlA014.html。

422　在日期选择器中显示年份下拉列表

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 datepicker()方法中设置参数 changeMonth 为 true、changeYear 为 true，从而实现在日期选择器中显示年份和月份的下拉列表框。当在 Google Chrome 浏览器中显示该页面时，单击“出生日期:”输入框，则将显示包含年份和月份下拉列表框的日期选择器，如图 422-1 所示。用户可以在日期选择器的任意年份和月份中进行选择。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $ (function() {
   $ ("#myDatepicker").datepicker({//允许在日期选择器中弹出月份和年份下拉列表框
     changeMonth: true, //在日期选择器中弹出月份下拉列表框
```

```
        changeYear: true//在日期选择器中弹出年份下拉列表框
    });});
   </script>
   <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
    /* 注意需要默认图标文件: images\ui - icons_777777_256x240.png */
    /* 注意需要默认图标文件: images\ui - icons_444444_256x240.png */
    /* 设置日期选择器的宽度 */
    #myDatepicker { width: 275px; }
   </style></head>
  <body><p>出生日期: <input type = "text" id = "myDatepicker"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myDatepicker"). datepicker({changeMonth: true, changeYear: true})中的 changeMonth 参数(属性)值为 true，表示在日期选择器中显示月份下拉列表框，在默认情况下日期选择器不显示月份的下拉列表框；changeYear 参数(属性)值为 true，表示在日期选择器中显示年份下拉列表框，在默认情况下日期选择器不显示年份的下拉列表框。如果仅设置 $("#myDatepicker"). datepicker({changeYear: true})，则在日期选择器中仅显示年份下拉列表框，不显示月份下拉列表框。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用 datepicker()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。此外，由于相关的图标源自默认图标文件 ui- icons_777777_256x240. png 和 ui-icons_444444_256x240. png(images 文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA170. html。



图 422-1

423 在日期选择器中显示月份下拉列表

此实例主要在 jquery-ui. min. js 和 jquery-ui. min. css 的 datepicker()方法中设置参数 changeMonth 为 true，从而实现在日期选择器中仅显示月份的下拉列表框。当在 Google Chrome 浏览器中显示该页面时，单击“出生日期:”输入框，则将显示包含月份下拉列表框的日期选择器，如图 423-1 所示。用户可以在日期选择器的任意月份中进行选择。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
   $(function() {//允许在日期选择器中弹出月份下拉列表框
     $("#myDatepicker").datepicker({changeMonth: true });});
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  /* 注意需要默认图标文件: images\ui - icons_777777_256x240.png */
  /* 注意需要默认图标文件: images\ui - icons_444444_256x240.png */
   #myDatepicker { width: 275px; } /* 设置日期选择器的宽度 */
```

```
</style></head>
<body><p>出生日期: <input type="text" id="myDatepicker"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myDatepicker").datepicker({changeMonth: true })中的changeMonth参数(属性)值为true，表示在日期选择器中显示月份下拉列表框，在默认情况下日期选择器不显示月份的下拉列表框。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用datepicker()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。此外，由于相关的图标源自默认图标文件ui-icons_777777_256x240.png和ui-icons_444444_256x240.png(images文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是myHtmlA169.html。

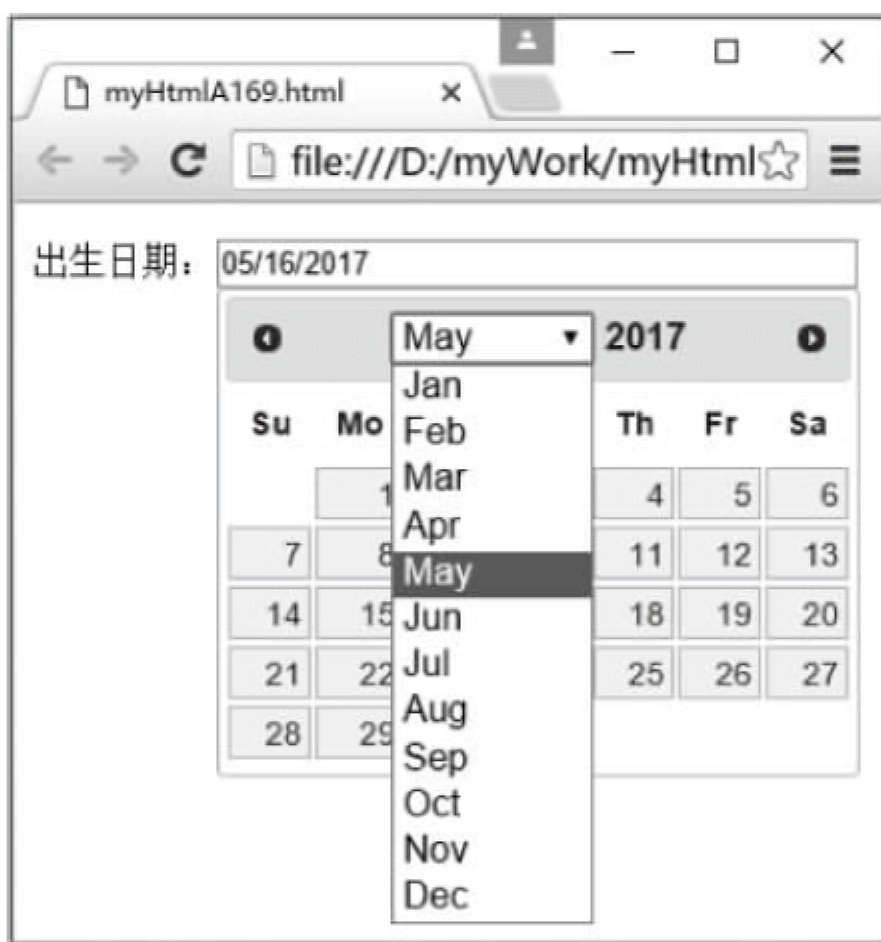


图　423-1

424　创建可显示和选择多月份的日期选择器

此实例主要在jquery-ui.min.js和jquery-ui.min.css的datepicker()方法中设置参数numberOfMonths，从而实现在日期选择器中显示多个月份供用户选择。当在Google Chrome浏览器中显示该页面时，单击"出生日期:"输入框，则显示5、6、7几个月份的日期选择器，如图424-1所示；单击日期选择器右上角的按钮，则显示6、7、8几个月份的日期选择器，如图424-2所示；单击日期选择器左上角的按钮，则显示4、5、6几个月份的日期选择器。用户可以在日期选择器的任意月份中进行选择。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {//显示3个月份的日期选择器
   $("#myDatepicker").datepicker({ numberOfMonths: 3 });
  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
```

```
    /＊注意需要默认图标文件：images\ui－icons_777777_256x240.png＊/
    /＊注意需要默认图标文件：images\ui－icons_444444_256x240.png＊/
    /＊设置日期选择器的宽度＊/
    ＃myDatepicker { width: 275px;}
  </style></head>
<body><p>出生日期：<input type="text" id="myDatepicker"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("＃myDatepicker").datepicker({ numberOfMonths：3 })中的 numberOfMonths 参数值用于指定需要显示的月份数量。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用 datepicker()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。此外，由于相关的图标源自默认图标文件 ui-icons_777777_256x240.png 和 ui-icons_444444_256x240.png(images 文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA168.html。

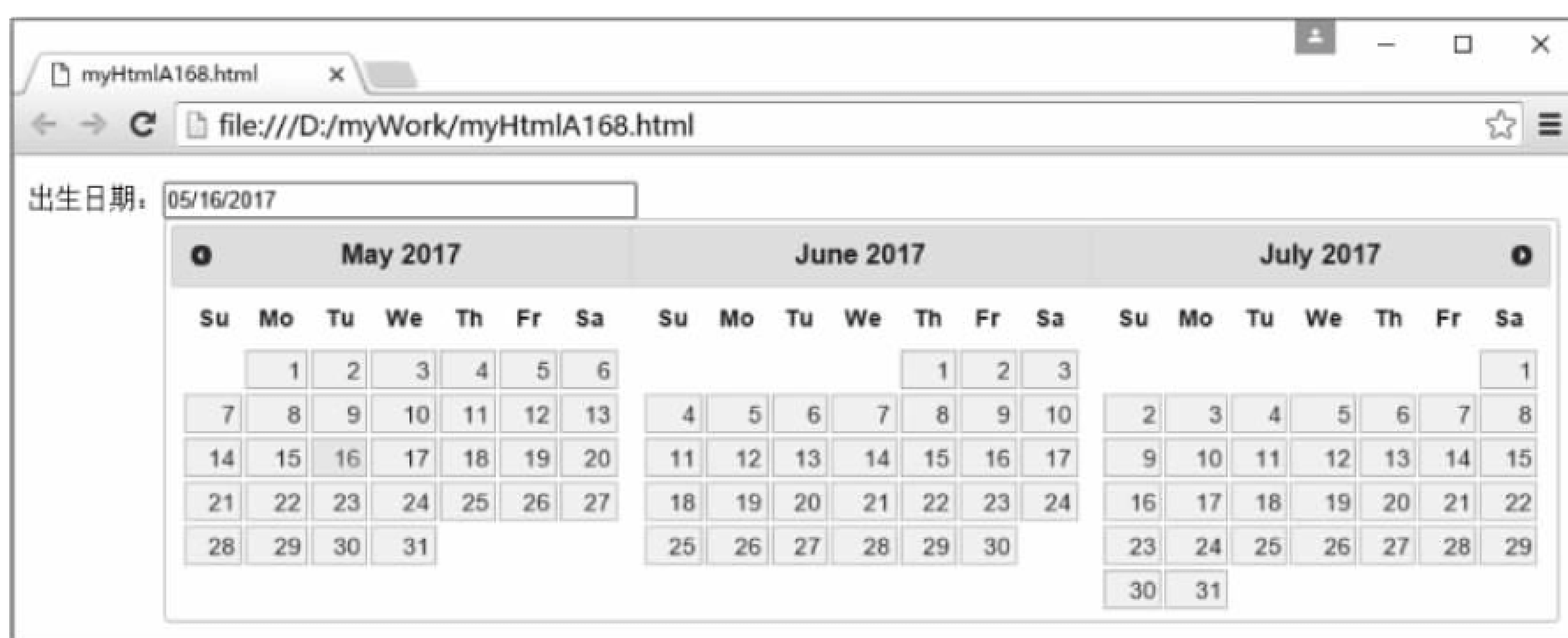


图 424-1

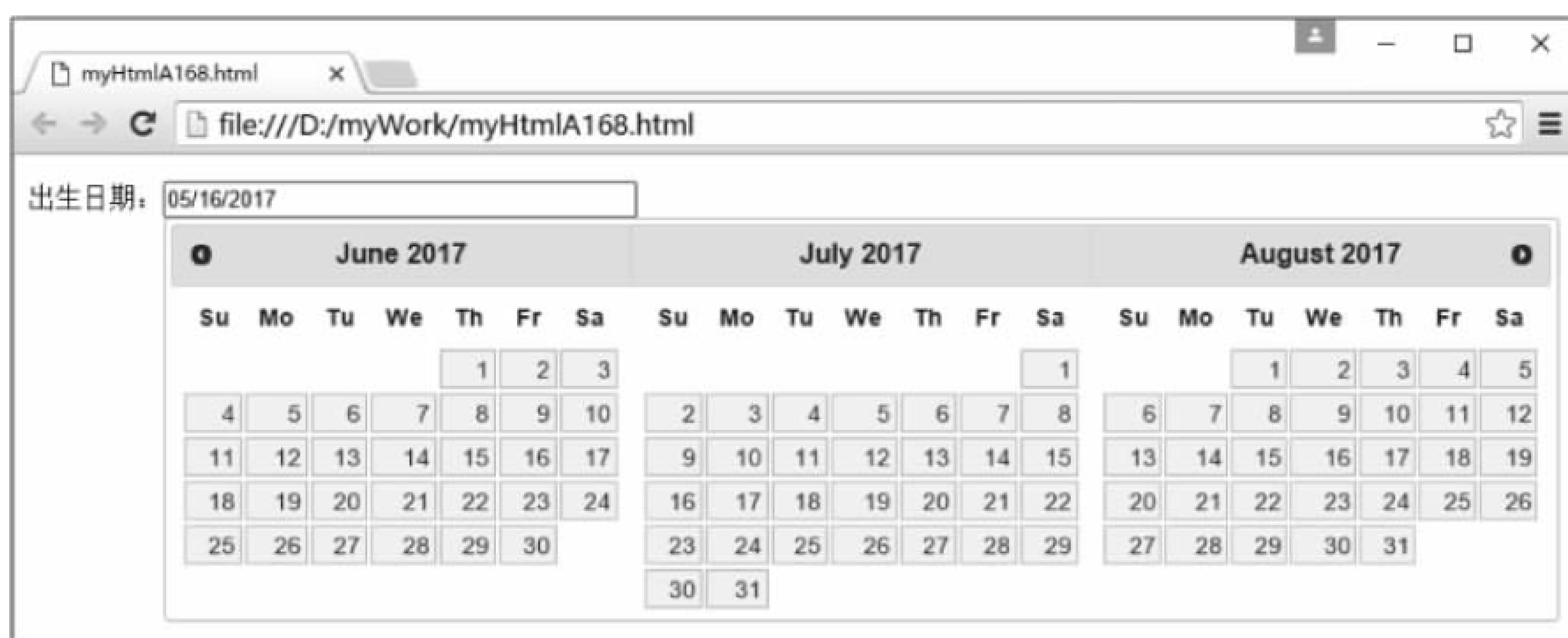


图 424-2

425 自动校验在范围内设置的年、月、日信息

此实例主要设置<input>标签的 type 属性为 datetime-local，从而实现限制用户在文本框中输入或选择有效的年、月、日、时、分等日期信息。当在浏览器中显示该页面时，在“考试日期：”文本框中将

显示预置的日期，用户可以直接在此修改年、月、日、时、分等日期信息，如图 425-1 所示。如果输入的月数大于 12，则被自动更正为 12；如果输入的日数大于 31，则被自动更正为 31；如果输入的小时数大于 23，则被自动更正为 23；如果输入的分钟数大于 59，则被自动更正为 59。单击“确认登记信息”按钮，则将在弹出的消息框中以纯文本的形式显示该日期信息。有关此实例的主要代码如下所示：

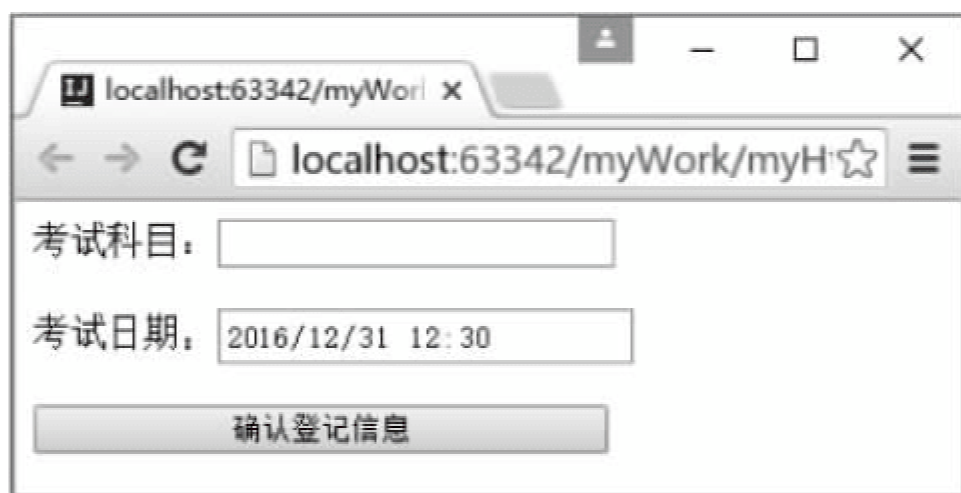


图　425-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {//确认登记信息
    alert("您登记的日期是: " + $("#myDate").prop("value"));
  })});
 </script></head>
<body><form style = "width: 400px;">考试科目: <input type = "text" id = "myName"/><br><br>考试日期:
<input type = "datetime - local" id = "myDate" value = "2016 - 12 - 07T12:30" /><br><br><button type =
"button" id = "myBtn" style = "width: 250px">确认登记信息</button></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，datetime-local 属性值是 HTML5 中 <input>标签的 type 属性的新值，该属性值能够使用户在文本框中选择或设置日期，可以通过<input>标签的 value 属性来获取该日期信息。

此实例的源文件名是 myHtmlA015.html。

426　自动校验用户设置的小时数和分钟数

此实例主要设置<input>标签的 type 属性为 time，从而实现限制用户在文本框中输入或选择有效的小时数和分钟数。当在浏览器中显示该页面时，在“考试时间：”文本框中将显示预置的小时数和分钟数，例如“09:30”，用户可以直接在此修改小时数和分钟数。如果输入的小时数大于 23，则被自动更正为 23；如果输入的分钟数大于 59，则被自动更正为 59。单击“确认登记信息”按钮，则将在弹出的消息框中以纯文本的形式显示该小时数和分钟数，如图 426-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script language = "javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {//确认登记信息
    alert("您登记的时间是: " + $("#myTime").prop("value"));
  }) });
 </script></head>
```

```
<body><form style="width: 400px;">考试科目:<input type="text" id="myName"/><br><br>考试时间:
<input type="time" id="myTime" value="09:30"/><br><br><button type="button" id="myBtn" style=
"width: 250px">确认登记信息</button></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,time 属性值是 HTML5 中<input>标签的 type 属性的新值,该属性值能够使用户在文本框中选择或设置时间(时、分),可以通过<input>标签的 value 属性来获取小时数和分钟数信息。

此实例的源文件名是 myHtmlA013.html。

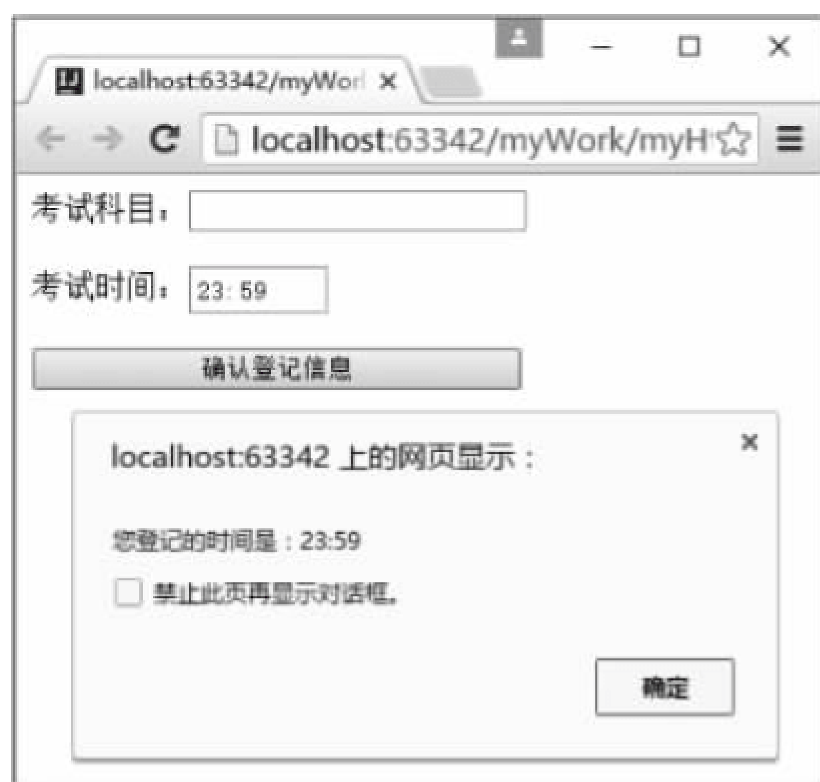


图 426-1

427 自定义在日期选择器中选择的日期格式

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 datepicker()方法中设置参数 option 为 dateFormat 的某一格式,从而实现根据指定的格式显示在日期选择器中选择的日期。当在 Google Chrome 浏览器中显示该页面时,如果设置"日期格式:"为"ISO 8601 - yy-mm-dd",则"出生日期:"日期选择器显示的日期是"2017-05-17",如图 427-1 所示;如果设置"日期格式:"为"With text - 'day' d 'of' MM 'in the year' yy",则"出生日期:"日期选择器显示的日期是"day 17 of May in the year 2017",如图 427-2 所示。有关此实例的主要代码如下。



图 427-1



图 427-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myDatepicker").datepicker({
    changeMonth: true, //在日期选择器中弹出月份下拉列表框
    changeYear: true //在日期选择器中弹出年份下拉列表框
   });
   $("#myFormat").change(function() {          //设置日期显示格式
     $("#myDatepicker").datepicker("option", "dateFormat", $(this).val());
  });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  #myDatepicker { width: 270px; }                /* 设置日期选择器的宽度 */
 </style></head>
<body><div align="center"><p>日期格式: <select id="myFormat">
   <option value="mm/dd/yy">Default - mm/dd/yy</option>
   <option value="yy-mm-dd">ISO 8601 - yy-mm-dd</option>
   <option value="d M, y">Short - d M, y</option>
   <option value="d MM, y">Medium - d MM, y</option>
   <option value="DD, d MM, yy">Full - DD, d MM, yy</option>
   <option value="&apos;day&apos; d &apos;of&apos; MM &apos;in the year&apos; yy">With text - 'day' d 'of'
MM 'in the year' yy</option>
  </select></p>
 <p>出生日期: <input type="text" id="myDatepicker"></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myDatepicker").datepicker("option", "dateFormat", $(this).val())中的$(this).val()为下拉列表框中某一选项具体值，例如“mm/dd/yy”，此行代码的作用就是设置日期选择器的日期格式为“mm/dd/yy”。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用datepicker()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。此外，由于相关的图标源自默认图标文件ui-icons_777777_256x240.png和ui-icons_444444_256x240.png(images文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是myHtmlA171.html。

428　在日期选择器中设置可选择的日期范围

此实例主要在jquery-ui.min.js和jquery-ui.min.css的datepicker()方法中设置参数minDate和maxDate，从而限制用户在日期选择器中可选的日期范围。当在Google Chrome浏览器中显示该页面时，用户只能选择2017-05-12到2017-05-22的日期，如图428-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {                          //限制可选日期为当前日期的前、后5天
   $("#myDatepicker").datepicker({ minDate: -5, maxDate: "5" });
  //下限为5天，上限为1个月零10天
  //$("#myDatepicker").datepicker({ minDate: -5,  maxDate: "+1M +10D"});
 });
```

```
</script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  #myDatepicker { width: 275px;}                    /* 设置日期选择器的宽度 */
 </style></head>
<body><p>出游日期: <input type = "text" id = "myDatepicker"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myDatepicker").datepicker({ minDate: -5, maxDate: "5" })表示用户在日期选择器中可选的日期范围为当前日期的前、后5天。其中，minDate和maxDate参数(选项)限制可选择的日期范围，如果设置为字符串，使用'D'表示天，使用'W'表示周，使用'M'表示月，使用'Y'表示年。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用datepicker()方法时必须添加jquery-ui.min.js和jquery-ui.min.css两个文件。此外，由于相关的图标源自默认图标文件ui-icons_777777_256x240.png和ui-icons_444444_256x240.png(images文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是myHtmlA172.html。

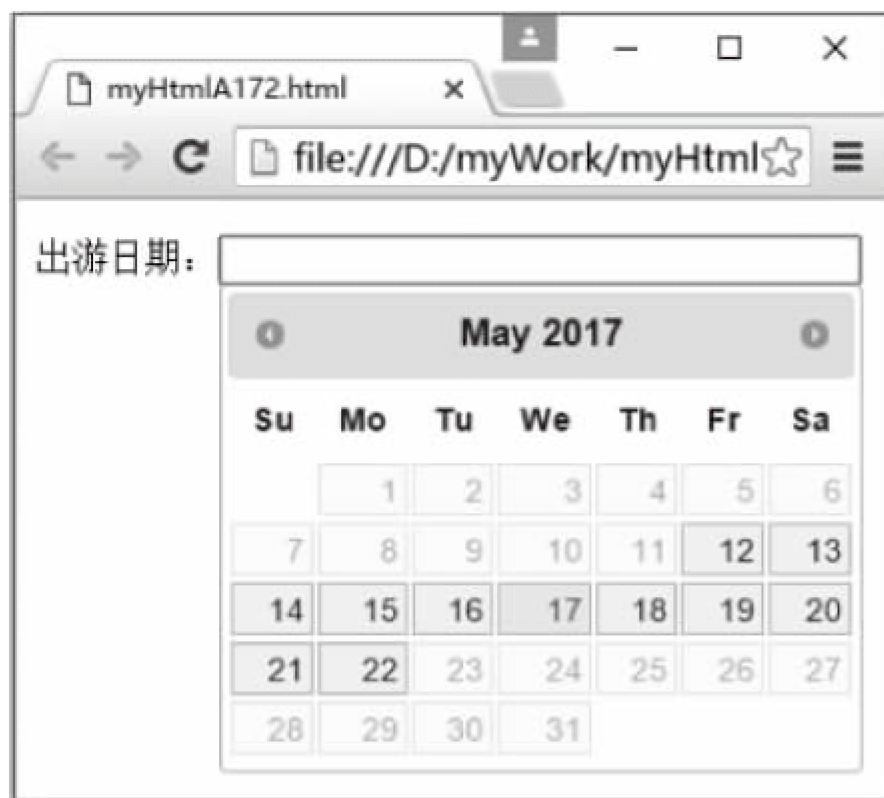


图 428-1

429 在日期选择器中禁止选择周末两天日期

此实例主要在jquery-ui.min.js和jquery-ui.min.css的datepicker()方法中设置参数beforeShowDay为$.datepicker.noWeekends，从而禁止用户选择日期选择器中的周末两天日期。当在Google Chrome浏览器中显示该页面时，星期六、星期天这两天日期处于灰色禁止状态，不可选择，如图429-1所示；单击标题栏左、右两端的"上一月""下一月"按钮切换到其他月份将出现相同的效果，如图429-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $(function() {                          //禁止选择日期选择器中的周末日期
  $("#myDatepicker").datepicker({beforeShowDay: $.datepicker.noWeekends });});
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  #myDatepicker { width: 275px; }          /* 设置日期选择器的宽度 */
```

```
</style></head>
<body><p>接待日期：<input type="text" id="myDatepicker"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#myDatepicker").datepicker({beforeShowDay：$.datepicker.noWeekends })用于禁止用户选择日期选择器中的星期六、星期天这两天日期。需要说明的是，datepicker()方法是日期选择器部件(Datepicker Widget)的方法，因此在使用 datepicker()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。此外，由于相关的图标源自默认图标文件 ui-icons_777777_256x240.png 和 ui-icons_444444_256x 240.png(images 文件夹下)，因此需要添加这些文件才能正确显示，否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA173.html。

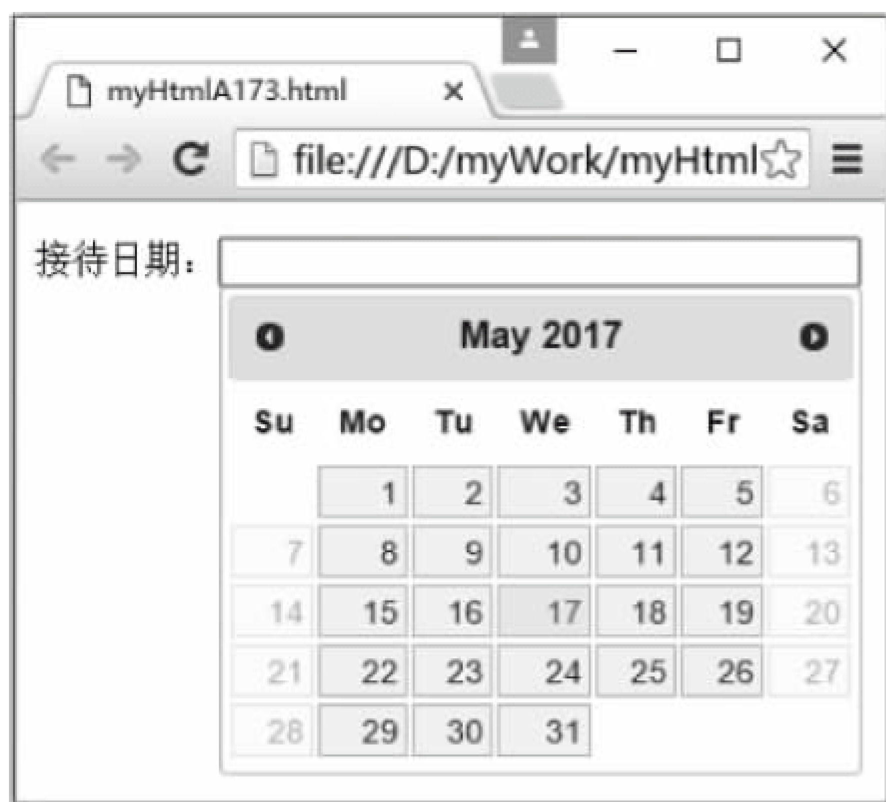


图　429-1

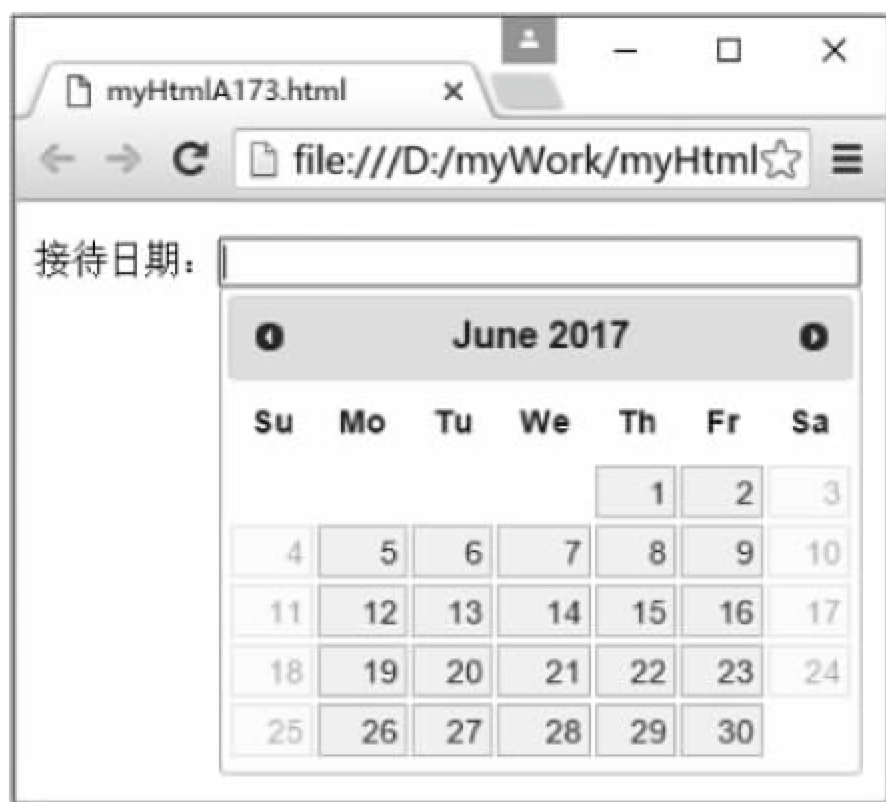


图　429-2

430　判断选择的日期在一年中处于第几周

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 的 $.datepicker.iso8601Week()方法判断用户选择的日期在一年中处于第几周。当在 Google Chrome 浏览器中显示该页面时，在“申报日期：”日期选择器中任意选择一个日期，例如“05/17/2017”，如图 430-1 所示；然后单击“显示周数”按钮，则将在弹出的消息框中显示周数信息，例如“05/17/2017 是当年第 20 周”，如图 430-2 所示。有关此实例的主要代码如下。

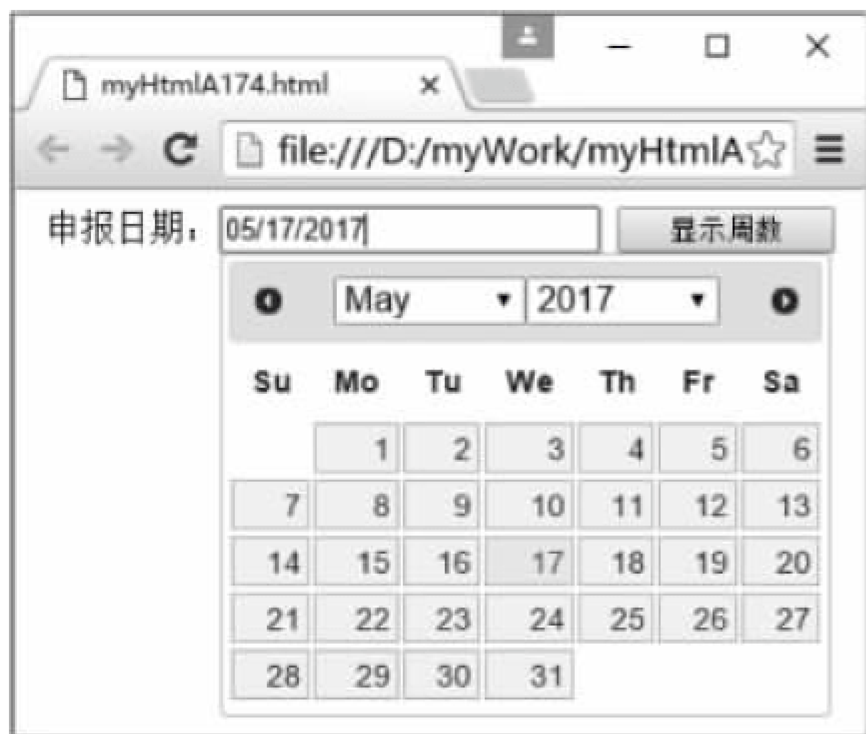


图　430-1

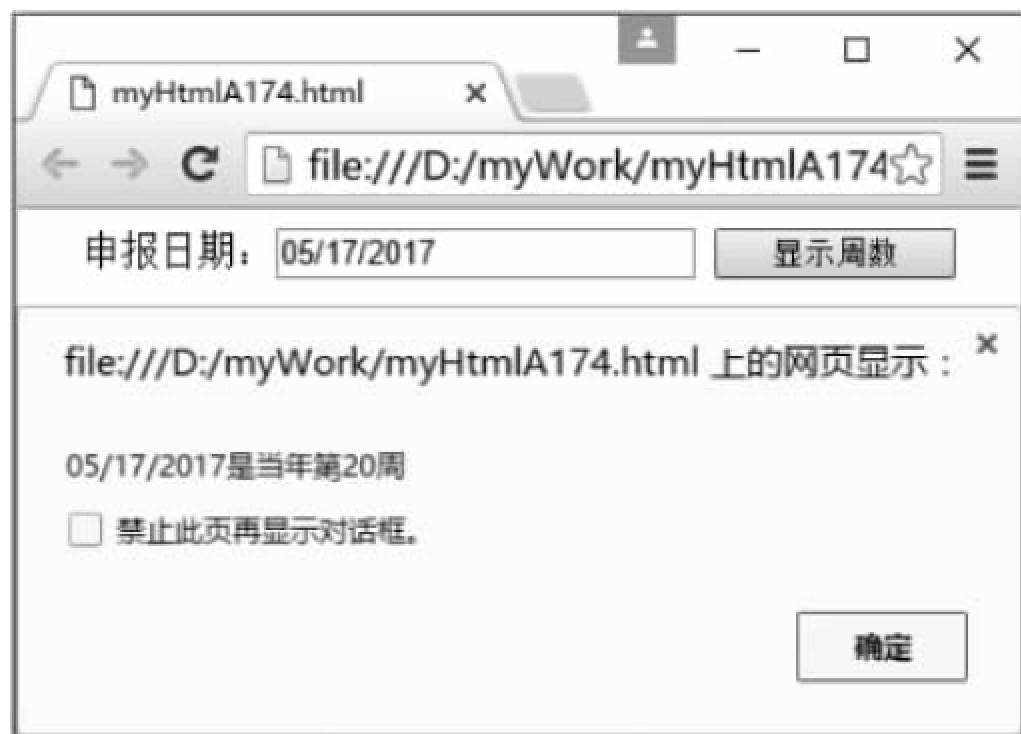


图　430-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myDatepicker").datepicker({ changeMonth: true,changeYear: true});
    $("#myBtnWeek").click(function() { //显示周数
     var myInfo=$("#myDatepicker").val()+"是当年第"+$.datepicker.iso8601Week( new Date( $("#
myDatepicker").val()))+"周";
     alert(myInfo);
  });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  #myDatepicker { width: 170px;}
  #myBtnWeek{ width:100px;}
 </style></head>
<body><div align="center">申报日期: <input type="text" id="myDatepicker">
 <button id="myBtnWeek">显示周数</button></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$.datepicker. iso8601Week(new Date($("#myDatepicker").val()))用于判断指定的日期在一年中处于第几周,该方法使用 ISO 8601 定义一周,一周从星期一开始,每一年的第一周都包含 1 月 4 日。这意味着上一年最多有 3 天可能包含在当年的第一周中,当年最多有 3 天可能包含在上一年的最后一周中。需要说明的是,iso8601Week()方法是日期选择器部件(Datepicker Widget)的方法,因此在使用 iso8601Week()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。此外,由于相关的图标源自默认图标文件 ui-icons_777777_256x240.png 和 ui-icons_444444_256x240.png(images 文件夹下),因此需要添加这些文件才能正确显示,否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA174.html。

431 实现在颜色选择器中选择颜色设置背景

此实例主要设置<input>标签的 type 属性为 color,从而实现允许用户在颜色选择器中选择颜色并设置页面的背景颜色。当在浏览器中显示该页面时颜色按钮将显示预置的颜色,单击此按钮将弹出颜色选择器,在颜色选择器中任意选择一种颜色,如图 431-1 所示,单击“确定”按钮,则颜色按钮将显示选择的颜色,然后单击“设置背景颜色”按钮,则使用选择的颜色设置页面的背景颜色,如图 431-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtn").click(function() {//设置背景颜色
     var myColor=$("#myColor").prop("value");
      $('body').css("background-color",myColor);
   }) });
 </script></head>
<body><form style="width: 400px;"> 请选择颜色: <input type="color" id="myColor" value="#00FF40" />
<button type="button" id="myBtn" style="width: 150px">设置背景颜色</button></form></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，color 属性值是 HTML5 中< input >标签的 type 属性的新值，该属性值能够使用户在颜色选择器中选择并定制颜色，用户可以通过< input >标签的 value 属性来获取颜色选择器返回的颜色值信息。

此实例的源文件名是 myHtmlA016. html。

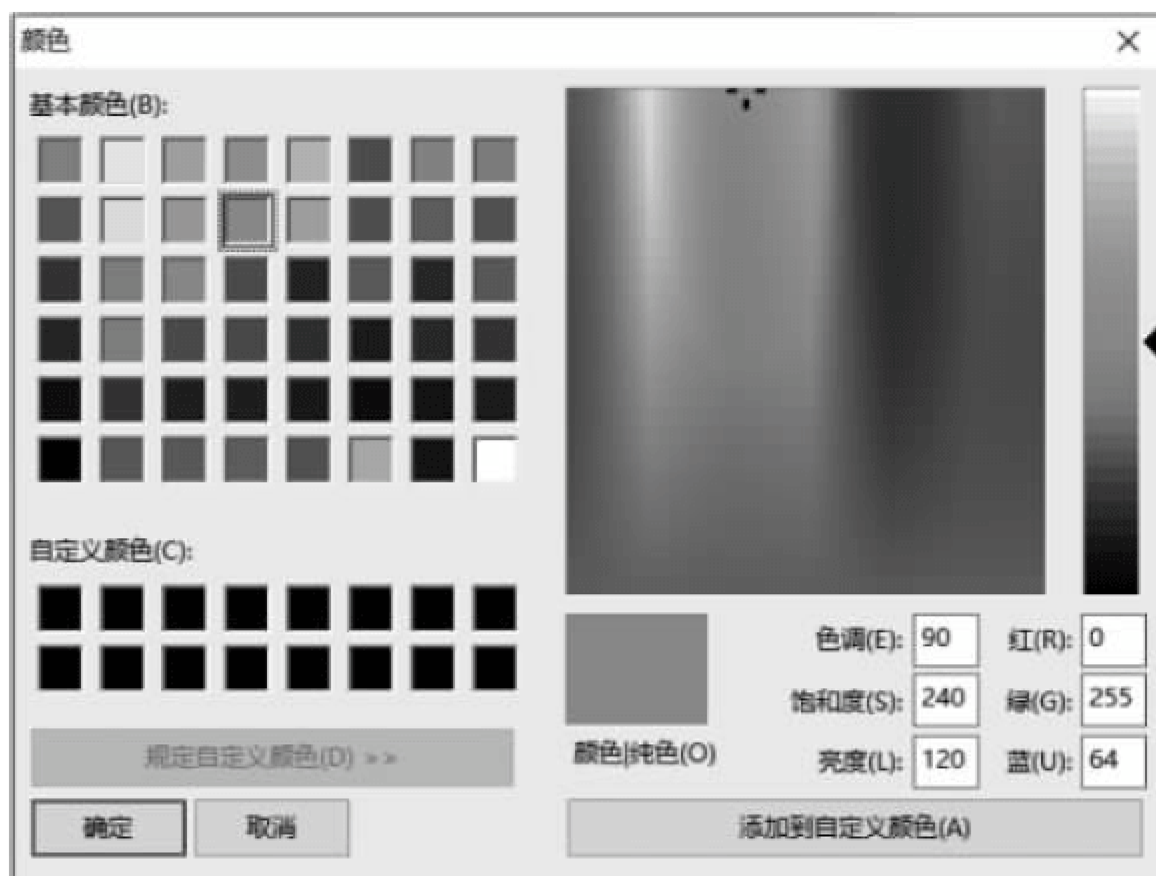


图 431-1



图 431-2

432 使用 radio 单选按钮实现纯 CSS 选项卡

此实例主要演示了如何使用 radio 单选按钮(input 元素)实现纯 CSS 的选项卡。当在 Google Chrome 浏览器中显示该页面时，单击"中央公园"选项卡，效果如图 432-1 所示；单击"山城夜景"选项卡，效果如图 432-2 所示；单击"乐山大佛"选项卡，效果如图 432-3 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8 >
 <style type = "text/css">
  /* 设置盒子的基本样式 */
  .myBox { width: 420px; min - height: 250px; position: relative; }
  /* 标题块空心部分 */
  .myTab {width: 25 % ; margin - right: - 1px; border: 1px solid #CCC; float: left;border - top - left -
        radius: 6px; border - top - right - radius: 6px;}
  /* 标题块实心部分 */
  . myLabel  {display:  block;  padding  -  top:  5px; padding  -  bottom:  5px;  background  -  color:
             lightgoldenrodyellow; text - align: center; border - top - left - radius: 6px; border - top
             - right - radius: 6px;}
  .myRadio, .myTab_content { position: absolute; left: - 4999em;
                            border - bottom - left - radius: 10px; border - bottom - right - radius: 10px;
                            border - top - right - radius: 10px; background - color: aqua; }
  /* 选中内容块 */
   .myRadio:checked ~ .myTab_content { margin - top: - 1px; padding: 1px;
       border: 1px solid #CCC;left: 0;right: 0; background - color: aqua; }
  /* 选中标题块 */
```

```
.myRadio:checked ~ .myLabel { border-bottom: 0px solid #FFF; position: relative; z-index: 1; margin-bottom: -1px; background-color: aqua; }
img {float: left; border-radius: 10px; padding: 5px; margin-top: 7px; margin-left: 7px; width: 210px;height: 140px; }
p { font-size: 14px; padding-left: 10px; padding-right: 10px;}
</style></head>
<body><div class="myBox"><div class="myTab">
<input type="radio" id="myTabRadio1" class="myRadio" name="tab" checked="checked"/>
<label class="myLabel" for="myTabRadio1">中央公园</label>
<div class="myTab_content"><img src="img/B179A.jpg"/><p>重庆中央公园是西南地区最大的开放式城市中心公园,西距在建的悦来重庆国际会展城3公里,东距江北国际机场5公里。它是一个依托重庆山水风貌特色,融会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一起,成为两江新区重大城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲去处,重庆的标志性公园。</p></div></div>
<div class="myTab"><input type="radio" id="myTabRadio2" class="myRadio" name="tab"/><label class="myLabel" for="myTabRadio2">山城夜景</label><div class="myTab_content">    <img src="img/B179B.jpg"/><p>山城夜景即重庆夜景。重庆市区三面临江,一面靠山,倚山筑城,建筑层叠耸起,道路盘旋而上,城市风貌独特,由此形成奇丽夜景。初夜山城,以繁华区灯饰群为中心,干道和桥梁华灯为纽带,万家民居灯火为背景,层见叠出,构成一片高下井然、错落有致、远近互衬的灯的海洋。车船流光,不停穿梭于茫茫灯海之中,依稀飞起汽笛、欢笑、笙歌之声,给夜中城平添无限生机。更兼两江波澄银树,浪卷金花,满天繁星似人间灯火,遍地华灯若天河群星,上下浑然一体,五彩交相辉映,如梦如幻,如诗如歌,堪足撩人耳目,动人心旌。不览夜景,未到重庆。</p></div></div>
<div class="myTab"><input type="radio" id="myTabRadio3" class="myRadio" name="tab"/><label class="myLabel" for="myTabRadio3">乐山大佛</label><div class="myTab_content"><img src="img/B118.jpg"/><p>乐山大佛,地处四川省乐山市,岷江、青衣江和大渡河三江汇流处,与乐山城隔江相望,北距成都160余公里。它是依凌云山栖霞峰临江峭壁凿造的一尊大佛,始凿于唐开元元年(公元713年),历时90余年方建成,建高71米,有"山是一尊佛,佛是一座山"之称,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态匀称,神势肃穆,依山凿成,临江危坐。大佛通高71米,头高14.7米,头宽10米,发髻1051个,耳长6.7米,鼻和眉长5.6米,嘴巴和眼长3.3米,颈高3米,肩宽24米,手指长8.3米,从膝盖到脚背28米,脚背宽9米,脚面可围坐百人以上。</p></div></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,checked选择器用于匹配每个已被选中的input元素,并且只用于单选按钮和复选框,在此实例中主要用于设置选中选项卡的CSS样式。

此实例的源文件名是myHtmlB179.html。



图 432-1



图　432-2



图　432-3

433　创建标签在左侧排列的纵向风格选项卡

此实例主要使用 jquery-ui. min. js 和 jquery-ui. min. css 的 tabs( )方法创建选项卡，并使用 addClass("ui-tabs-vertical")方法添加 CSS 样式，从而实现选项卡的标签纵向分布在内容的左侧。当在 Google Chrome 浏览器中显示该页面时，如果单击左侧的标签“乐山大佛”，则将显示“乐山大佛”对应的内容，如图 433-1 所示；如果单击左侧的标签“中央公园”，则将显示“中央公园”对应的内容，如图 433-2 所示。有关此实例的主要代码如下。



图　433-1



图　433-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {                          //创建标签在左侧的纵向风格布局的选项卡
 $("#tabs").tabs().addClass("ui-tabs-vertical"); });
```

```
</script>
<link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  #tabs { border: none; width: 461px; }                    /*设置选项卡的基本样式*/
  /*设置标签盒子的基本样式*/
  .mybox { background - color: white; border: none;padding: 0 !important;}
  /*设置选项卡内容部分的基本样式*/
  #tabs - 1, #tabs - 2 { border: 0px solid gray; border - radius: 10px; width: 320px;
  background - color: cyan; padding: 5px 15px; }
  /*设置标签的基本样式*/
  .ui - tabs - vertical .ui - tabs - nav { float: left; width: 100px; margin - top: 50px; }
  /*设置标签列表项的基本样式*/
  .ui - tabs - vertical .ui - tabs - nav li {width: 100%;
   border - bottom - width: 1px !important; border - right - width: 0px  !important;
   border - bottom - left - radius: 5px; border - top - left - radius: 5px;  }
  /*设置标签超链接的基本样式*/
  .ui - tabs - vertical .ui - tabs - nav li a { display: block;outline: none;}
  /*设置焦点标签的基本样式*/
  .ui - tabs - vertical .ui - tabs - nav li.ui - tabs - active {
          padding - bottom: 0;border:none !important; background - color: cyan; }
  /*设置内容面板的基本样式*/
  .ui - tabs - vertical .ui - tabs - panel { padding - left:10px; float: left; }
  /*设置图像的基本样式*/
  img { float: left; border - radius: 10px; padding: 5px; width: 210px; height: 135px;
box - shadow: 2px 2px 10px black; margin - right: 12px; margin - top: 17px; margin - bottom: 5px; margin -
left: 2px; }
 </style></head>
<body><div id = "tabs"><ul class = "mybox">
  <li><a href = "#tabs - 1">乐山大佛</a></li>
  <li><a href = "#tabs - 2">中央公园</a></li></ul>
 <div id = "tabs - 1"><img src = "img/B118.jpg" style = "float:left;"><p>乐山大佛,地处四川省乐山市,岷
江、青衣江和大渡河三江汇流处,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态
匀称,神势肃穆,依山凿成,临江危坐。大佛通高 71 米,头高 14.7 米,头宽 10 米,发髻 1051 个,耳长 6.7 米,鼻和
眉长 5.6 米,嘴巴和眼长 3.3 米,颈高 3 米,肩宽 24 米,手指长 8.3 米,从膝盖到脚背 28 米,脚背宽 9 米,脚面可
围坐百人以上。</p></div>
 <div id = "tabs - 2"><img src = "img/B179A.jpg" style = "float:left;"><p>重庆中央公园是西南地区最大
的开放式城市中心公园,西距在建的悦来重庆国际会展城 3 公里,东距江北国际机场 5 公里。它是一个依托重
庆山水风貌特色,融会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一
起,成为两江新区重大城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲
去处,重庆的标志性公园。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#tabs").tabs(). addClass("ui-tabs-vertical")用于通过设置 CSS 样式将选项卡的标签重置于内容的左侧,在使用 addClass("ui-tabs-vertical")方法实现将标签布局在选项卡的左侧之前,通常需要像实例这样设置. ui-tabs-vertical 下面的相关元素样式。tabs()方法主要用于实现单击选项卡的标签页,切换到标签页对应的内容,该方法是选项卡部件(Tabs Widget)的方法,因此在使用 tabs()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA144. html。

434 创建在标签右侧包含“关闭”按钮的选项卡

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 的 tabs()方法创建默认的选项卡，并使用 delegate()方法处理，单击标签右侧的“关闭”按钮事件，从而实现单击选项卡标签右侧的“关闭”按钮即可关闭该标签页。当在 Google Chrome 浏览器中显示该页面时将显示两个在标签右侧包含“关闭”按钮的选项卡，如图 434-1 所示；单击标签“乐山大佛”右侧的“关闭”按钮，则将关闭“乐山大佛”标签页，如图 434-2 所示；单击标签“中央公园”右侧的“关闭”按钮将实现类似的功能。有关此实例的主要代码如下。



图 434-1



图 434-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    var tabs = $( "#tabs" ).tabs();                //创建默认的选项卡
    //"关闭"按钮,当单击时移除标签页
     $("#tabs").delegate( "span.ui-icon-close", "click", function() {
      var panelId = $( this ).closest( "li" ).remove().attr( "aria-controls" );
       $( "#" + panelId ).remove();
      tabs.tabs( "refresh" ); });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  #tabs { border: none;   width: 400px; }
  /* 设置标签盒子的基本样式 */
  .mybox { border: none; padding-left: 10px !important; }
  #tabs-1, #tabs-2 { border: 0px solid gray; border-radius: 5px; background-color: cyan; }
  /* 设置标签导航栏列表项的基本样式 */
  .ui-tabs-nav li { border: none;display: block; background-color: gray!important;
                  border-top-right-radius: 5px; border-top-left-radius: 5px;
                  border: none !important; }
```

```
/*设置标签导航栏焦点列表项的基本样式*/
.ui-tabs-nav li.ui-tabs-active { margin-bottom: -1px; padding-bottom: 1px;
                         background-color: cyan!important; font-weight: bold; }
/*设置标签列表项超链接的基本样式*/
.ui-tabs-nav li a { display: block; outline: none; color: navy!important; }
/*设置图像的基本样式*/
img {float: left; border-radius: 10px; padding: 5px; width: 210px; height: 135px;
    box-shadow: 2px 2px 10px black; margin-right: 12px; margin-top: 17px; margin-bottom: 5px;
    margin-left: 2px; }
/*设置"关闭"按钮的基本样式*/
#tabs li .ui-icon-close {background:url("img/A146.png") no-repeat;
         background-size: 15px 15px;background-position:1px 2px;
         float: left; margin: 0.4em 0.2em 0 0; cursor: pointer;}
</style></head>
<body><div id="tabs"><ul class="mybox">
 <li><a href="#tabs-1">乐山大佛</a><span class="ui-icon ui-icon-close"></span></li>
 <li><a href="#tabs-2">中央公园</a><span class="ui-icon ui-icon-close"></span></li></ul>
<div id="tabs-1"><img src="img/B118.jpg" style="float:left;"><p>乐山大佛,地处四川省乐山市,岷
江、青衣江和大渡河三江汇流处,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态
匀称,神势肃穆,依山凿成,临江危坐。大佛通高 71 米,头高 14.7 米,头宽 10 米,发髻 1051 个,耳长 6.7 米,鼻和
眉长 5.6 米,嘴巴和眼长 3.3 米,颈高 3 米,肩宽 24 米,手指长 8.3 米,从膝盖到脚背 28 米,脚背宽 9 米,脚面可
围坐百人以上。</p></div>
<div id="tabs-2"><img src="img/B179A.jpg" style="float:left;"><p>重庆中央公园是西南地区最大
的开放式城市中心公园,西距在建的悦来重庆国际会展城 3 公里,东距江北国际机场 5 公里。它是一个依托重
庆山水风貌特色,融会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一
起,成为两江新区重大城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲
去处,重庆的标志性公园。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,#tabs li .ui-icon-close { }用于设置“关闭”按钮的基本样式;用于将“关闭”按钮放置在标签上;$("#tabs").tabs()用于创建一个默认的选项卡;$("#tabs").delegate("span.ui-icon-close","click",function(){})用于处理鼠标单击“关闭”按钮事件。需要说明的是,tabs()方法是选项卡部件(Tabs Widget)的方法,因此在使用 tabs()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA146.html。

435 创建通过底部的标签进行导航的选项卡

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 的 tabs()方法创建默认的选项卡,并使用$(".ui-tabs-nav").appendTo(".tabs-bottom")方法添加 CSS 样式,从而实现将选项卡的标签从顶部移动到底部。当在 Google Chrome 浏览器中显示该页面时,如果单击底部的标签“乐山大佛”,则将显示“乐山大佛”对应的内容,如图 435-1 所示;如果单击底部的标签“中央公园”,则将显示“中央公园”对应的内容,如图 435-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() { $("#tabs").tabs();                //创建选项卡
```

```
    $(".ui-tabs-nav").appendTo(".tabs-bottom");        //移动导航标签到底部
                        });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   #tabs { border: none; width: 400px; }                    /*设置选项卡的基本样式*/
   /*设置选项卡内容部分的基本样式*/
   #tabs-1, #tabs-2 { border: 0px solid gray; border-radius: 10px; width: 100%;
                  background-color: cyan; padding: 5px 15px; }
   /*设置标签盒子的基本样式*/
   .mybox { background-color: white; border: none; padding: 0 !important; }
   /*设置标签导航栏的基本样式*/
   .tabs-bottom .ui-tabs-nav { clear: left; margin-left:110px; }
   /*设置标签导航栏列表项的基本样式*/
   .tabs-bottom .ui-tabs-nav li { top: auto;  bottom: 0; margin: 0 0.2em 1px 0; border-top: 0;
display: block; background-color: gray; border-bottom-right-radius: 5px; border-bottom-left-
radius: 5px; }
   /*设置标签导航栏焦点列表项的基本样式*/
   .tabs-bottom .ui-tabs-nav li.ui-tabs-active { font-weight: bold; margin-top: -1px; padding
-top: 1px; border:none !important;background-color: cyan;}
   /*设置标签列表项超链接的基本样式*/
   .tabs-bottom .ui-tabs-nav li a {display: block; outline: none; color: navy; }
   /*设置图像的基本样式*/
   img { float: left; border-radius: 10px; padding: 5px; width: 210px; height: 135px;
box-shadow: 2px 2px 10px black; margin-right: 12px; margin-top: 17px; margin-bottom: 5px;  margin-
left: 2px; }
 </style></head>
<body><div id="tabs" class="tabs-bottom">
 <ul class="mybox"><li><a href="#tabs-1">乐山大佛</a></li>
  <li><a href="#tabs-2">中央公园</a></li></ul>
 <div id="tabs-1"><img src="img/B118.jpg" style="float:left;"><p>乐山大佛,地处四川省乐山市,岷
江、青衣江和大渡河三江汇流处,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态
匀称,神势肃穆,依山凿成,临江危坐。大佛通高 71 米,头高 14.7 米,头宽 10 米,发髻 1051 个,耳长 6.7 米,鼻和
眉长 5.6 米,嘴巴和眼长 3.3 米,颈高 3 米,肩宽 24 米,手指长 8.3 米,从膝盖到脚背 28 米,脚背宽 9 米,脚面可
围坐百人以上。</p></div>
 <div id="tabs-2"><img src="img/B179A.jpg" style="float:left;"><p>重庆中央公园是西南地区最大
的开放式城市中心公园,西距在建的悦来重庆国际会展城 3 公里,东距江北国际机场 5 公里。它是一个依托重
庆山水风貌特色,融会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一
起,成为两江新区重大城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲
去处,重庆的标志性公园。</p></div></div></body></html>
```



图 435-1



图 435-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#tabs").tabs()用于创建一个默认风格的选项卡，$(".ui-tabs-nav").appendTo(".tabs-bottom")用于将默认风格的选项卡的导航标签从顶部移动到底部。当将默认风格的选项卡的导航标签从顶部移动到底部时，通常需要像实例这样重新设置.tabs-bottom、.ui-tabs-nav 等类的 CSS 样式。需要说明的是，tabs()方法是选项卡部件(Tabs Widget)的方法，因此在使用 tabs()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA145.html。

436　实现鼠标指针悬浮在选项卡标签上时切换对应内容

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 tabs()方法中传递{event:"mouseover"}参数，从而实现鼠标指针悬浮在选项卡标签上时切换对应的内容。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在“乐山大佛”标签上，则将显示对应的内容，如图 436-1 所示；如果鼠标指针悬浮在“中央公园”标签上，也将显示对应的内容，如图 436-2 所示。有关此实例的主要代码如下。



图　436-1



图　436-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() { //鼠标指针悬浮在选项卡标签上时切换对应的内容
   $( "#tabs" ).tabs({ event: "mouseover" }); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  .mybox { background-color: white; border: none; }
  #tabs { border: none; width: 400px; }
  #tabs-1, #tabs-2 { border: 1px solid gray; border-radius: 5px; }
  /*设置图像的基本样式*/
```

```
  img {float: left; border-radius: 10px; padding: 5px; width: 210px;height: 135px;
box-shadow: 2px 2px 10px black; margin-right: 12px; margin-top:17px; margin-bottom: 5px; margin-left:2px; }
 </style></head>
<body><div id="tabs">
 <ul class="mybox"><li><a href="#tabs-1">乐山大佛</a></li>
                   <li><a href="#tabs-2">中央公园</a></li></ul>
 <div id="tabs-1"><img src="img/B118.jpg" style="float:left;"><p>乐山大佛,地处四川省乐山市,岷江、青衣江和大渡河三江汇流处,是世界上最大的石刻大佛。乐山大佛头与山齐,足踏大江,双手抚膝,大佛体态匀称,神势肃穆,依山凿成,临江危坐。大佛通高 71 米,头高 14.7 米,头宽 10 米,发髻 1051 个,耳长 6.7 米,鼻和眉长 5.6 米,嘴巴和眼长 3.3 米,颈高 3 米,肩宽 24 米,手指长 8.3 米,从膝盖到脚背 28 米,脚背宽 9 米,脚面可围坐百人以上。</p></div>
 <div id="tabs-2"><img src="img/B179A.jpg" style="float:left;"><p>重庆中央公园是西南地区最大的开放式城市中心公园,西距在建的悦来重庆国际会展城 3 公里,东距江北国际机场 5 公里。它是一个依托重庆山水风貌特色,融会中西文化的现代城市公园,将成为重庆新地标。开园后,将与年底投用的国际会展城一起,成为两江新区重大城市功能板块。这是国际中心区率先启动的双核,这里也将成为重庆新中心最大的休闲去处,重庆的标志性公园。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#tabs").tabs({event:"mouseover"})用于鼠标指针悬浮在选项卡标签上时切换对应的内容,如果此方法没有{event:"mouseover"}参数,直接调用$("#tabs").tabs(),则选项卡的内容只有在鼠标单击标签时才会进行切换。tabs()方法主要用于实现单击选项卡的标签页切换到标签页对应的内容,该方法是选项卡部件(Tabs Widget)的方法,因此在使用 tabs()方法时必须添加 jquery- ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA143.html。

437 以拖曳方式在选项卡中增加、移除选项

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 中的 sortable()方法,从而实现以拖曳的方式将选项卡的一个标签列表中的选项移动到另一个标签列表中。当在 Google Chrome 浏览器中显示该页面时,“数学基础教材”标签中的列表项如图 437-1 所示,“计算机专业教材”标签中的列表项如图 437-2 所示。使用鼠标拖动“数学基础教材”标签列表中的“常微分方程”列表项到“计算机专业教材”标签,如图 437-3 所示;然后松开鼠标,则“常微分方程”列表项就会被插入“计算机专业教材”标签的列表中,如图 437-4 所示。改变之后的“数学基础教材”标签中的列表项如图 437-5 所示。反之亦然。有关此实例的主要代码如下。



图 437-1



图 437-2



图 437-3



图 437-4



图 437-5

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
  //拖曳列表项到选项卡的标签,从而实现把列表项从一个标签移到另一个标签
  $( "#sortable1, #sortable2" ).sortable().disableSelection();
   var $tabs = $( "#tabs" ).tabs();
   var $tab_items = $( "ul:first li", $tabs ).droppable({
        accept: ".connectedSortable li",
        hoverClass: "ui-state-hover",
        drop: function( event, ui ) {
               var $item = $( this );
               var $list = $( $item.find( "a" ).attr( "href" ) )
                              .find( ".connectedSortable" );
               ui.draggable.hide( "slow", function() {
                 $tabs.tabs( "option", "active", $tab_items.index( $item ) );
                 $(this ).appendTo( $list ).show( "slow" );
```

```
  });} });});
 </script>
 <link rel = "stylesheet" href = "css/jquery - ui.min.css"><style type = "text/css">
  .mybox{background - color: white; border: none; }
  #tabs - 1, #tabs - 2{border: 1px solid gray;border - radius: 5px;}
  #tabs{ border:none; width:400px;}
  /* 设置列表项的基本样式 */
   #sortable1 li, #sortable2 li { margin: 0 5px 5px 5px; padding: 5px; font - size: 1.2em; width: 180px;
}
  /* 设置列表项超链接的基本样式 */
  #sortable1 li a, #sortable2 li a { - webkit - transition: 0.3s ease - out; background: lightgreen;
color: black; padding: 7px 15px 7px 10px; border - radius: 5px; width: 150px;display: block; text -
decoration: none;font - size: 14px; - webkit - box - shadow: 2px 2px 4px #888; }
 </style></head>
<body><div id = "tabs" >
 <ul class = "mybox"><li><a href = "#tabs - 1">数学基础教材</a></li>
                     <li><a href = "#tabs - 2">计算机专业教材</a></li></ul>
 <div id = "tabs - 1" ><ul id = "sortable1" class = "connectedSortable ui - helper - reset ">
    <li><a href = "javascript:void(0)">数学分析</a></li>
    <li><a href = "javascript:void(0)">常微分方程</a></li>
    <li><a href = "javascript:void(0)">复变函数</a></li></ul></div>
 <div id = "tabs - 2"><ul id = "sortable2" class = "connectedSortable ui - helper - reset">
    <li><a href = "javascript:void(0)">数据结构</a></li>
    <li><a href = "javascript:void(0)">数据库系统概论</a></li>
    <li><a href = "javascript:void(0)">操作系统设计与实现</a></li>
    <li><a href = "javascript:void(0)">离散数学</a></li></ul></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(function(){})内部的代码用于实现拖曳一个标签中的列表项到选项卡的另一个标签，从而实现把一个标签中的列表项移动到另一个标签中。sortable()方法是 jQuery UI 的可排序小部件(Sortable Widget)的方法，在任意的 DOM 元素上启用 sortable 功能，通过鼠标单击并拖曳元素到列表中的一个新的位置，其他条目就会自动调整。在默认情况下，sortable 的各个条目共享 draggable 属性。注意，在使用 sortable()方法实现拖动排序前必须添加 jquery-ui. min.js 和 jquery-ui. min.css 两个文件。

此实例的源文件名是 myHtmlA135.html。

438 使用方向键在表格的输入框间跳转

此实例主要通过添加表格的 keydown 事件的自定义方法实现使用键盘上的上、下、左、右方向键在表格单元格内置的多个输入框之间跳转。当在 Google Chrome 浏览器中显示该页面时即可使用键盘上的上、下、左、右方向键在表格单元格内置的多个输入框之间跳转，如果使用左方向键，当到达每行的左端时自动跳转到上一行右端的输入框；如果使用右方向键，当到达每行的右端时自动跳转到下一行左端的输入框，如图 438-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
  var myInputs = document.getElementsByTagName("input");
  function keyDown(event) {
   var myFocus = document.activeElement;
```

```
    if (!document.contains(myFocus)) return;
    var myEvent = window.event || event;
    var myKey = myEvent.keyCode;
    for (var i = 0; i < myInputs.length; i++) {
      if (myInputs[i] === myFocus) break;
    }
    switch (myKey) {//处理上、下、左、右方向键跳转
     case 37://方向键(←): VK_LEFT (37)
      if (i > 0) myInputs[i - 1].focus(); break;
     case 38://方向键(↑): VK_UP (38)
      if (i - 3 >= 0) myInputs[i - 3].focus(); break;
     case 39://方向键(→): VK_RIGHT (39)
      if (i < myInputs.length - 1) myInputs[i + 1].focus(); break;
     case 40://方向键(↓): VK_DOWN (40)
      if (i + 3 < myInputs.length) myInputs[i + 3].focus(); break;
  } }
 </script>
 <style type="text/css">
   * { font-size: 14px; text-align: center; }
  table, tr, td { border: 1px solid green; }
  td { width: 130px; }
  input { width: 70px; }
 </style></head>
<body><form><table onkeydown="keyDown(event)">
   <tr><td>数据列 1 </td><td>数据列 2 </td><td>数据列 3 </td></tr>
   <tr><td>数据 1-1:<input type="text" name="myData1-1" autofocus="on"/></td>
       <td>数据 1-2:<input type="text" name="myData1-2"/></td>
       <td>数据 1-3:<input type="text" name="myData1-3"/></td></tr>
   <tr><td>数据 2-1:<input type="text" name="myData2-1"/></td>
       <td>数据 2-2:<input type="text" name="myData2-2"/></td>
       <td>数据 2-3:<input type="text" name="myData2-3"/></td></tr>
   <tr><td>数据 3-1:<input type="text" name="myData3-1"/></td>
       <td>数据 3-2:<input type="text" name="myData3-2"/></td>
       <td>数据 3-3:<input type="text" name="myData3-3"/></td></tr>
   <tr><td>数据 4-1:<input type="text" name="myData4-1"/></td>
       <td>数据 4-2:<input type="text" name="myData4-2"/></td>
       <td>数据 4-3:<input type="text" name="myData4-3"/></td></tr>
   <tr><td>数据 5-1:<input type="text" name="myData5-1"/></td>
       <td>数据 5-2:<input type="text" name="myData5-2"/></td>
       <td>数据 5-3:<input type="text" name="myData5-3"/></td></tr>
   <tr><td>数据 6-1:<input type="text" name="myData6-1"/></td>
       <td>数据 6-2:<input type="text" name="myData6-2"/></td>
       <td>数据 6-3:<input type="text" name="myData6-3"/></td></tr></table>
</form></body></html>
```



图 438-1

上面有底纹的代码是此实例的核心代码。在该部分代码中，document.activeElement 属性返回文档中当前获得焦点的元素，如果需要为元素设置焦点，可以使用 element.focus()方法，也可以使用 document.hasFocus()方法来查看当前元素是否获取焦点；var myKey=myEvent.keyCode 用于获取当前按键的键值，并据此在 switch(myKey)中确定上、下、左、右方向键应该执行的动作。

此实例的源文件名是 myHtmlA116.html。

439　使用固定算法等距分配表格的列宽

此实例主要设置表格的 table-layout 属性为 fixed，从而实现等距分配表格的列宽。当在 Google Chrome 浏览器中显示该页面时，单击"等距分配列宽"按钮，则表格等距分配列宽的效果如图 439-1 所示；单击"自动调整列宽"按钮，则表格自动调整列宽的效果如图 439-2 所示。有关此实例的主要代码如下。



图　439-1



图　439-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   var myData=[{"HTML5 从入门到精通":"清华大学出版社"},
               {"HTML5+CSS3 从入门到精通":"清华大学出版社"},
               {"HTML5 经典实例":"中国电力出版社"},
               {"HTML5 入门经典":"机械工业出版社"},
               {"深入浅出 HTML5 编程":"东南大学出版社"}];
   for (var i in myData) {
    for (var key in myData[i]) {
      $("table").append("<tr><td>"+key+"</td><td>"+myData[i][key]+"</td></tr>");
  } }
  $("button#fixed").click(function() {       //等距分配列宽
   $("table").css("table-layout","fixed");
  });
  $("button#normal").click(function() {      //自动调整列宽
   $("table").css("table-layout","auto");
  });});
 </script>
 <style type="text/css">
  table, table tr td { border: 1px solid darkgreen; border-collapse: collapse; text-align: center; font
                      -size: 14px;  padding: 3px; table-layout:auto; }
   button{ width:190px; margin: 5px;}
```

```
</style></head>
<body><div align="center">
 <button id="fixed">等距分配列宽</button>
 <button id="normal">自动调整列宽</button>
 <table width="400px"><tr><td>图书名称</td><td>出版社</td></tr></table></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，table-layout:auto 表示元素(table)采用默认的自动算法计算表格每列的列宽。在 CSS3 中，table-layout 属性用于设置或检索表格的布局算法，该属性包含两个属性值——fixed 和 auto。auto 表示使用默认的自动算法，布局将基于各单元格的内容，换而言之，可能给某个单元格定义宽度为 100px，但结果可能并不是 100px，并且需要表格在每一单元格读取计算之后才会显示出来，速度很慢；fixed 表示使用固定布局的算法。在该算法中水平布局仅仅基于表格的宽度、表格边框的宽度、单元格间距、列的宽度，而和表格内容无关，也就是说内容可能被裁切。

此实例的源文件名是 myHtmlB361.html。

440 使用省略号代替固定表格列宽的长字符

此实例主要通过设置表格单元格的 width、table-layout、text-overflow、overflow 等属性实现使用省略号代替固定表格列宽的超长字符。当在 Google Chrome 浏览器中显示该页面时将自动根据单元格的字符串长度确定表格的列宽，单击"省略号代替超长字符"按钮，则表格的第 2 行和第 4 行由于书名的长度超出设置的列宽，将在末尾以省略号代替超长的部分，如图 440-1 所示；单击"自动调整列宽"按钮，则表格自动调整列宽，效果如图 440-2 所示。有关此实例的主要代码如下。



图 440-1

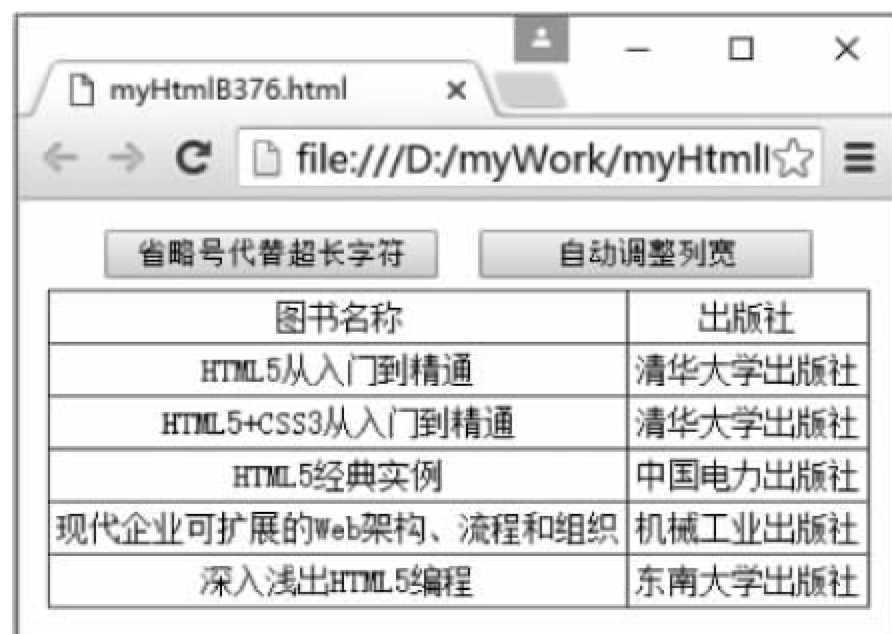


图 440-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {//加载表格数据
   var myData=[{"HTML5 从入门到精通": "清华大学出版社"},
               {"HTML5+CSS3 从入门到精通": "清华大学出版社"},
               {"HTML5 经典实例": "中国电力出版社"},
               {"现代企业可扩展的 Web 架构、流程和组织": "机械工业出版社"},
               {"深入浅出 HTML5 编程": "东南大学出版社"}];
  for (var i in myData) {
   for (var key in myData[i]) {
```

```
        $ ("table").append("<tr><td>" + key + "</td><td>" + myData[i][key] + "</td></tr>");   } }
     $ ("#myBtnEllipsis").click(function() {     //省略号代替超长字符
      $ ("table,table tr td").each(function() {
       $ (this)[0].className = "myEllipsis";  });});
     $ ("#myBtnNormal").click(function() {      //自动调整列宽
      $ ("table,table tr td").each(function() {
       $ (this)[0].className = "myAutomatic";  });});});
   </script>
   <style type = "text/css">
    /* 设置表格的基本样式 */
    table, table tr td { border: 1px solid darkgreen; border - collapse: collapse;
                         text - align: center; font - size: 14px; padding: 3px; table - layout: auto; white -
                         space: nowrap; }
    .myAutomatic { table - layout: auto;}          /* 自动设置列宽 */
    /* 设置列宽并用省略号代替超长字符 */
    .myEllipsis { width: 150px; table - layout: fixed;
                  text - overflow: ellipsis; overflow: hidden; }
    button { width: 145px; margin: 5px;}           /* 设置按钮的基本样式 */
   </style></head>
  <body><div align = "center">
   <button id = "myBtnEllipsis">省略号代替超长字符</button>
   <button id = "myBtnNormal">自动调整列宽</button>
   <table><tr><td>图书名称</td><td>出版社</td></tr></table></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，实现使用省略号代替固定表格列宽的超长字符主要使用了 width、table-layout、text-overflow、overflow 几个属性，width:150px 表示表格每列的宽度是 150px；table-layout：fixed 表示列宽由表格宽度和列宽度设定；text-overflow：ellipsis 表示显示省略号来代表被修剪的文本；overflow：hidden 表示单元格中的超长内容会被修剪，并且其余内容是不可见的。

此实例的源文件名是 myHtmlB376.html。

441　在表格的标题及单元格上添加阴影效果

此实例主要在 CSS 样式中设置 table、td、th 元素的 box-shadow 属性值，从而实现在表格的标题及单元格上添加阴影效果。当在 Google Chrome 浏览器中显示该页面时，在表格的标题及单元格上添加的阴影效果如图 441-1 所示。有关此实例的主要代码如下。



葡萄酒名称	生产日期	原产地	葡萄品种
罗博克葡萄酒	2015. 12. 6	西班牙	歌海娜
科洛文红葡萄酒	2015. 10. 5	法国	丹魄
马蒂尔葡萄酒	2014. 6. 9	意大利	朗布罗斯科

图　441-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  table{ border-spacing:10px;box-shadow:0px 0px 8px 7px gray;
         background-color: lightcyan; margin: 20px;}
  td,th{ background-color: lightcyan;box-shadow:5px 5px 6px gray;padding:10px;}
 </style></head>
<body><table>
 <thead><tr><th>葡萄酒名称</th><th>生产日期</th>
        <th>原产地</th><th>葡萄品种</th></tr></thead>
 <tr><td>罗博克葡萄酒</td><td>2015.12.6</td>
        <td>西班牙</td><td>歌海娜</td></tr>
 <tr><td>科洛文红葡萄酒</td><td>2015.10.5</td>
        <td>法国</td><td>丹魄</td></tr>
 <tr><td>马蒂尔葡萄酒</td><td>2014.6.9</td>
        <td>意大利</td><td>朗布罗斯科</td></tr></table></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow:0px 0px 8px 7px gray 用于设置表格的阴影样式，两个 0px 分别代表阴影的水平偏移量和垂直偏移量，8px 表示阴影的模糊半径，7px 表示阴影的扩散半径，gray 表示阴影的颜色；box-shadow:5px 5px 6px gray 用于设置表格标题栏 th 和单元格 td 的阴影样式，两个 5px 分别代表阴影的水平偏移量和垂直偏移量，6px 表示阴影的模糊半径，gray 表示阴影的颜色。

此实例的源文件名是 myHtmlB054.html。

442 使用 JSON 传递的天气数据自动填充表格

此实例主要实现使用 jQuery 的 ajax()方法查询远程数据，并通过远程传递的 JSON 格式的天气数据自动填充表格。当在 Google Chrome 浏览器中显示该页面时，单击“获取用户 IP 值并根据 IP 查询当地天气信息”按钮，则将自动从远程服务器查询到当地的天气信息，并显示在表格中，如图 442-1 所示；单击“清除数据”按钮，则将清除表格中的所有数据。有关此实例的主要代码如下。



图 442-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myBtnGetdata").click(function() {  //获取用户 IP 值并根据 IP 查询当地天气信息
```

```
$ ("table").html("");
var url = 'http://chaxun.1616.net/s.php?type = ip&output = json&callback = ?&_ = ' + Math.random();
$ .getJSON(url, function (data) {   var myCity = data.Isp;
 $ ("p#myLocation").text("您所在的位置: " + myCity.substr(0, 3));
 $ .ajax({
    url: "http://api.map.baidu.com/telematics/v3/weather",
    type: "get",
    data: {location: myCity.substr(0, 3),output: 'json',
           ak: '6tYzTvGZSOpYB5Oc2YGGOKt8' },
    dataType: "jsonp",
    success: function (data) {                    //获取天气数据
      var myData = data.results[0].weather_data;
      for (var i = 0; i < myData.length; i++) {
         $ ("table").append("<tr></tr>");
      }
      $ ("table tr").each(function (index) {
       $ (this).append("<td>" + myData[index].date + "</td><td><img src = " + myData[index].
dayPictureUrl + "/></td><td><img src = " + myData[index]. nightPictureUrl + "/></td><td>" + myData
[index].temperature + "</td><td>" + myData[index].weather + "</td><td>" + myData[index].wind +
"</td>");
   });} }) });});
  $ ("#myBtnClear").click(function(){              //清除数据
   $ ("table").html("");
 });});
</script>
<style type = "text/css">
  #myBtnGetdata { width: 350px; }
  #myBtnClear { width: 130px;}
</style></head>
<body><div align = "center">
 <button id = "myBtnGetdata">获取用户 IP 值并根据 IP 查询当地天气信息</button>
 <button id = "myBtnClear">清除数据</button>
 <p id = "myLocation"></p><table></table></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,ajax()方法用于通过 HTTP 请求加载远程数据,该方法是 jQuery 底层 AJAX 实现,简单易用的高层实现有 $.get()、$.post()等方法。$.ajax()方法返回其创建的 XMLHttpRequest 对象。在大多数情况下无须直接操作该函数,除非需要操作不常用的选项,以获得更多的灵活性。在最简单的情况下,$.ajax()可以不带任何参数直接使用。下面是此实例已经使用的 ajax()方法的参数的相关说明。

(1) url:表示发送请求的地址。

(2) type:表示请求方式,例如 post 或 get,默认为 get。

(3) data:要求为 Object 或 String 类型的参数,发送到服务器的数据。如果已经不是字符串,将自动转换为字符串格式。在 get 请求中将附加在 url 后。为防止这种自动转换,可以查看 processData 选项。对象必须为 key/value 格式,例如{foo1:"bar1",foo2:"bar2"}转换为 &foo1=bar1&foo2=bar2。如果是数组,jQuery 将自动为不同值对应同一个名称,例如{foo:["bar1","bar2"]}转换为 &foo=bar1&foo=bar2。

(4) dataType:要求为 String 类型的参数,预期服务器返回的数据类型。如果不指定,jQuery 将自动根据 http 包 mime 信息返回 responseXML 或 responseText,并作为回调函数参数传递。其可用的类型如下。

① xml：返回 XML 文档，可用 jQuery 处理。

② html：返回纯文本 HTML 信息，包含的 script 标签会在插入 DOM 时执行。

③ script：返回纯文本 JavaScript 代码，不会自动缓存结果，除非设置了 cache 参数。注意在远程请求时(不在同一个域下)所有 post 请求都将转为 get 请求。

④ json：返回 JSON 数据。

⑤ jsonp：JSONP 格式，在使用 JSONP 形式调用函数时，例如 myurl? callback=?，jQuery 将自动替换后一个"?"为正确的函数名，以执行回调函数。

⑥ text：返回纯文本字符串。

(5) success：要求为 Function 类型的参数，请求成功后调用的回调函数，有两个参数，一是由服务器返回，并根据 dataType 参数进行处理后的数据；二是描述状态的字符串。

此实例的源文件名是 myHtmlA175.html。

443　以模态方式显示对话框并获取其返回值

此实例主要实现使用< dialog >标签创建对话框，并通过 dialog 元素的 showModal()方法以模态方式显示此对话框。当在浏览器中显示该页面时，单击"显示对话框"按钮，则将以模态方式显示弹出的对话框，如图 443-1 所示；在模态对话框中单击"是"按钮，则将在消息框中显示单击按钮的返回值，如图 443-2 所示。有关此实例的主要代码如下。

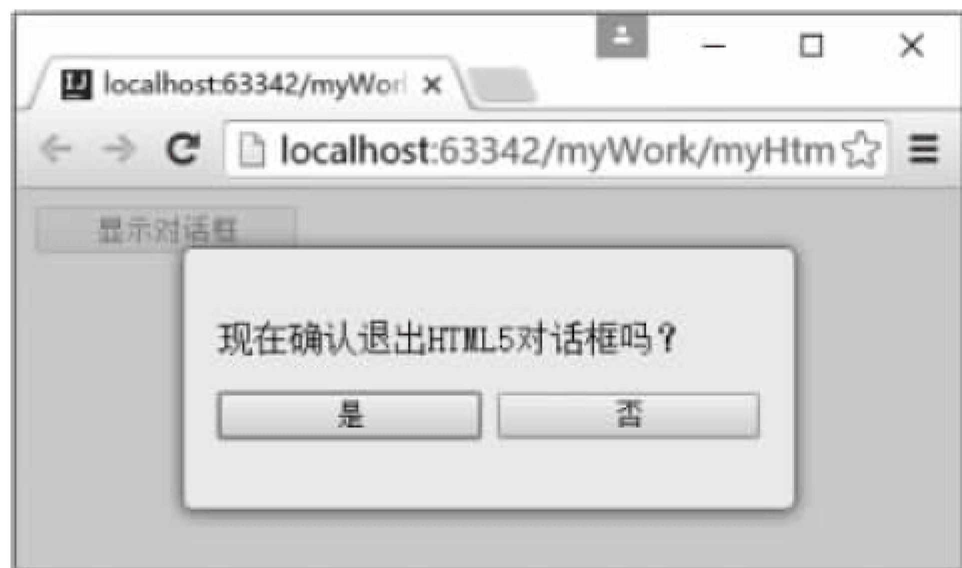


图　443-1

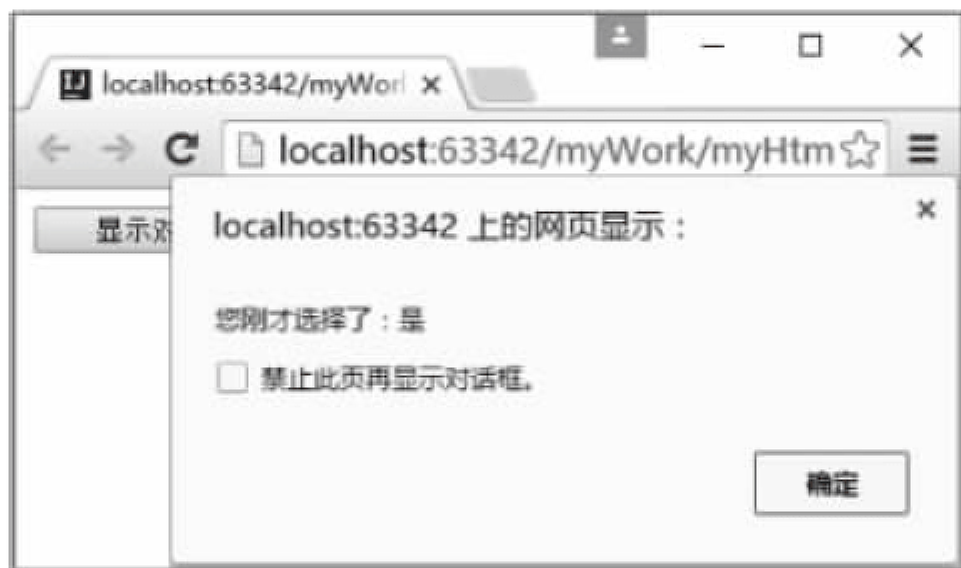


图　443-2

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    $("#myBtn").click(function() {          //显示对话框
     $("#myDlg").get(0).showModal();});
  myClose = function() {                     //响应关闭对话框
     alert("您刚才选择了: " + $("#myDlg").get(0).returnValue); } });
 </script>
 <style>
  dialog {background: #F0F0F0; border: 1px solid gray; border - radius: 5px;box - shadow: 0 1px 10px gray;}
  dialog::backdrop {background: rgba(173, 216, 230, 0.7);}
  button{ width:120px;}
 </style></head>
<body><button id = "myBtn">显示对话框</button>
<dialog id = "myDlg" onclose = "myClose()">
 <form method = "dialog"><p>现在确认退出 HTML5 对话框吗?</p>
```

```
    <p align = "center"><button type = "submit" value = "是">是</button>
   <button type = "submit" value = "否">否</button></p></form></dialog></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<dialog>标签是 HTML5 中的新标签，该标签对应 dialog 元素。在使用<dialog>标签创建对话框时通常需要操作 dialog 元素的下列属性、方法及事件。

dialog 元素的 open 属性用于设置其可见性，当其值为 true 时 dialog 元素可见，为 false 时不可见。dialog 元素的 returnValue 属性表示对话框被关闭时的返回值，当 dialog 元素的 close()方法被调用并传入参数时，returnValue 属性值将被设置为 close()方法的参数值。

dialog 元素的 show()方法用于显示对话框，此时页面上的其他元素仍可交互。dialog 元素的 showModal()方法则能够以模态方式显示对话框，此时页面上的其他元素不可交互，与 show()方法不同的是，如果 dialog 元素的 open 属性值为 true，调用此方法将报错。dialog 元素的 close()方法用于关闭(隐藏)对话框，该方法接受一个可选参数，如果传入参数，则 dialog 元素的 returnValue 属性将会被设置为参数值，如果 dialog 元素的 open 属性值为 false 或未设置，调用此方法将报错。

dialog 元素的 close 事件在关闭 dialog 元素时触发。dialog 元素的 cancel 事件在按键盘的 Esc 键关闭模态对话框时触发。

此实例的源文件名是 myHtmlA005.html。

444 创建包含“确认”和“取消”按钮的模式对话框

此实例主要在 jquery-ui.min.js 和 jquery-ui.min.css 的 dialog()方法中设置 modal 参数为 true，从而创建包含“确认”和“取消”按钮的模式对话框。当在 Google Chrome 浏览器中显示该页面时，单击“显示模式对话框”按钮，则将显示模式对话框，此时页面的其他部分变为灰色，不可操作，如图 444-1 所示。有关此实例的主要代码如下。

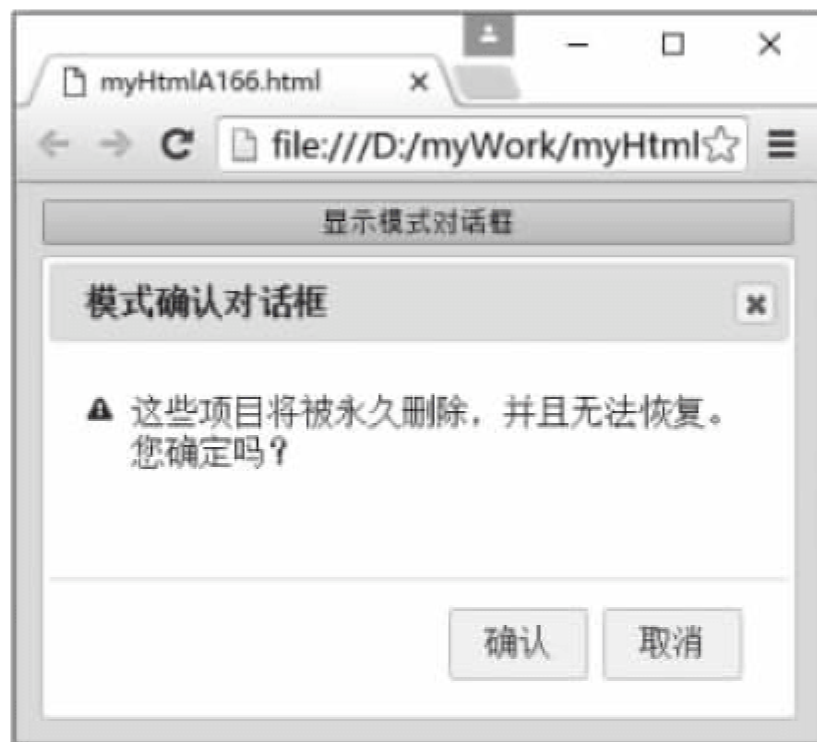


图 444-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $(function() {
   $("#myBtnDlg").click(function(){     //显示模式对话框
```

```
    $("#myDlg").dialog({resizable: false, height:220,width:350, modal: true,buttons: { "确认":
function() { $( this ).dialog( "close" ); },"取消": function() { $( this ).dialog( "close" ); }
  } });});});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  #myBtnDlg { width: 355px; }              /* 设置显示自定义可拖曳对话框按钮的基本样式 */
  #myDlg { display: none;}                 /* 初始时隐藏模式对话框 */
 </style></head>
<body><div align="center"><button id="myBtnDlg">显示模式对话框</button></div>
<div id="myDlg" title="模式确认对话框">
 <p><span class="ui-icon ui-icon-alert" style="float:left; margin:0 7px 20px 0;"></span>这些项
目将被永久删除,并且无法恢复。您确定吗?</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myDlg").dialog({resizable: false,height:220,width:350,modal:true,buttons:{"确认":function(){$(this).dialog("close");},"取消":function(){$(this).dialog("close");}}})用于设置模式对话框的属性值,例如禁止用户拖动边框线改变尺寸、宽度和高度,以及在对话框中添加按钮等,其中最重要的是设置 modal:true,这是区分模式对话框和非模式对话框的参数。

模式(modal)对话框即模态对话框,是指用户在页面中想要对对话框以外的内容进行操作时必须首先对该对话框进行响应,例如单击"确定"或"取消"按钮等将该对话框关闭,与之相对应的就是非模式对话框。需要说明的是,dialog()方法是对话框部件(Dialog Widget)的方法,因此在使用 dialog()方法时必须添加 jquery-ui.min.js 和 jquery-ui. min.css 两个文件。此外,由于对话框的图标源自默认图标文件 ui-icons_777777_256x240.png 和 ui-icons_444444_256x240.png(images 文件夹下),因此需要添加这些文件才能正确显示,否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA166.html。

445 创建可自定义标题栏的"关闭"按钮的对话框

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 的 dialog()方法创建带"关闭"按钮的可自由拖曳的对话框。当在 Google Chrome 浏览器中显示该页面时,单击"显示自定义可拖曳对话框"按钮,则将显示自定义可拖曳对话框,此时即可拖动自定义对话框的灰色标题栏任意移动,也可以单击右上角的"关闭"按钮或下面的"关闭自定义可拖曳对话框"按钮关闭该对话框,如图 445-1 所示。有关此实例的主要代码如下。



图 445-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {
    $("#myBtnDlg").click(function() {           //显示自定义可拖曳对话框
     $("#myDlg").dialog(); });
    $("#myBtnClose").click(function() {         //关闭自定义可拖曳对话框
     $("#myDlg").dialog("close"); });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
 <style type="text/css">
    * { -webkit-user-select: none; }
   .ui-dialog{ width:350px !important; }        /* 设置自定义可拖曳对话框的基本样式 */
   /* 设置自定义可拖曳对话框标题栏的基本样式 */
   .ui-dialog-title { text-align: center; color: navy; }
    #myBtnDlg{ width:355px; }                   /* 设置"显示自定义可拖曳对话框"按钮的基本样式 */
    #myBtnClose{ width:250px; }                 /* 设置"关闭自定义可拖曳对话框"按钮的基本样式 */
    #myDlg{ display: none; }                    /* 初始时隐藏可拖曳对话框 */
 </style></head>
<body><div align="center">
 <button id="myBtnDlg">显示自定义可拖曳对话框</button></div>
<div id="myDlg" title="自定义可拖曳对话框">
 <p>jQuery UI 是以 jQuery 为基础的开源 JavaScript 网页用户界面代码库。包含底层用户交互、动画、特效和
可更换主题的可视控件。我们可以直接用它来构建具有很好交互性的 Web 应用程序。所有插件测试能兼容 IE
6.0+、Firefox 3+、Safari 3.1+、Opera 9.6+和 Google Chrome。</p>
 <div align="center"><button id="myBtnClose">关闭自定义可拖曳对话框</button> </div></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，dialog()方法用于显示默认的对话框，此对话框由一个标题栏和一个内容区域组成，并且可以移动、调整尺寸，默认可通过右上角的“关闭”按钮关闭。dialog("close")用于关闭默认的对话框。需要说明的是，dialog()方法是对话框部件(Dialog Widget)的方法，因此在使用 dialog()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。此外，由于对话框的图标源自默认图标文件 ui-icons_777777_256x240.png(images 文件夹下)，因此需要添加此文件才能显示，否则在图标位置会显示空白。

此实例的源文件名是 myHtmlA165.html。

446 为超链接创建自定义风格的工具提示框

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 的 tooltip()方法实现为超链接创建自定义风格的工具提示框。当在 Google Chrome 浏览器中显示该页面时，如果将鼠标指针悬浮在“量子力学”超链接上，则显示的自定义工具提示框效果如图 446-1 所示；如果将鼠标指针悬浮在“测不准原理”超链接上，则显示的自定义工具提示框效果如图 446-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
   $(function() {                                   //创建自定义风格的提示框
```

```
    $(document).tooltip({
      position: { my: "center bottom-20",     at: "center top",
                 using: function (position, feedback) {
      $(this).css(position); $("<div>").addClass("arrow"). addClass(feedback. vertical).addClass
(feedback.horizontal).appendTo(this);
  } } });});
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
   /* 设置提示框和箭头的背景及边框线 */
   .ui-tooltip, .arrow:after { background: black; border: 2px solid white;}
   /* 设置提示框的基本样式 */
   .ui-tooltip {padding: 10px 20px; color: white; border-radius: 20px;
                font: bold 14px "Helvetica Neue", Sans-Serif;
                text-transform: uppercase; box-shadow: 0 0 7px black; }
   /* 设置箭头的基本样式 */
   .arrow { width: 70px; height: 16px; overflow: hidden;  position: absolute;
           left: 50%; margin-left: -35px; bottom: -16px; }
   .arrow.top { top: -16px; bottom: auto; }          /* 设置箭头顶端的基本样式 */
   .arrow.left {left: 20%; }                         /* 设置箭头左端的基本样式 */
   /* 将正方形旋转 45°使之成为箭头 */
   .arrow:after { content: ""; position: absolute; left: 20px; top: -20px; width: 25px; height: 25px;
                box-shadow: 6px 5px 9px -9px black; -webkit-transform: rotate(45deg); }
   .arrow.top:after { bottom: -20px;  top: auto;} /* 调整箭头顶端的样式 */
   /* 设置盒子的基本样式 */
   . myBox{ display: inline-block; width: 450px; padding: 15px;   margin: 10px; background-color:
            #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0,
            0.06) inset; position: relative; border-radius: 5px; }
   body { background-color: lightgray; }
 </style></head>
<body><div class="myBox"><p>有关现实存在的真正意义,长期以来是一个热烈讨论的问题。直到<a href
="http://baike.so.com/doc/2737738-2889635.html" title="量子力学是研究微观粒子的运动规律的物理
学分支学科,它提供粒子"似-粒"、"似-波"双重性(即"波粒二象性")及能量与物质相互作用的数学描述。">量
子力学</a>,以及海森堡的<a href="http://baike.so.com/doc/6631264-7603760.html" title="粒子的位
置与动量不可同时被确定">测不准原理</a>的出现,争议才在于是否我们所观察到的事物存在于我们的心智之
外,或是如同理想主义者所认为的是我们心智感知之物。根据这个宇宙的概念,近来的实验皆显示出,现实境况
除非是在观测之下,否则实际上可能并不存在。一项由澳洲国立大学研究人员所进行的波粒二象性研究中指
出,粒子或波在它们被观测之后呈现存在的状态。换句话说,观测就是一切,而现实的境况只有在观测发生时才
会存在。</p></div>
</body></html>
```

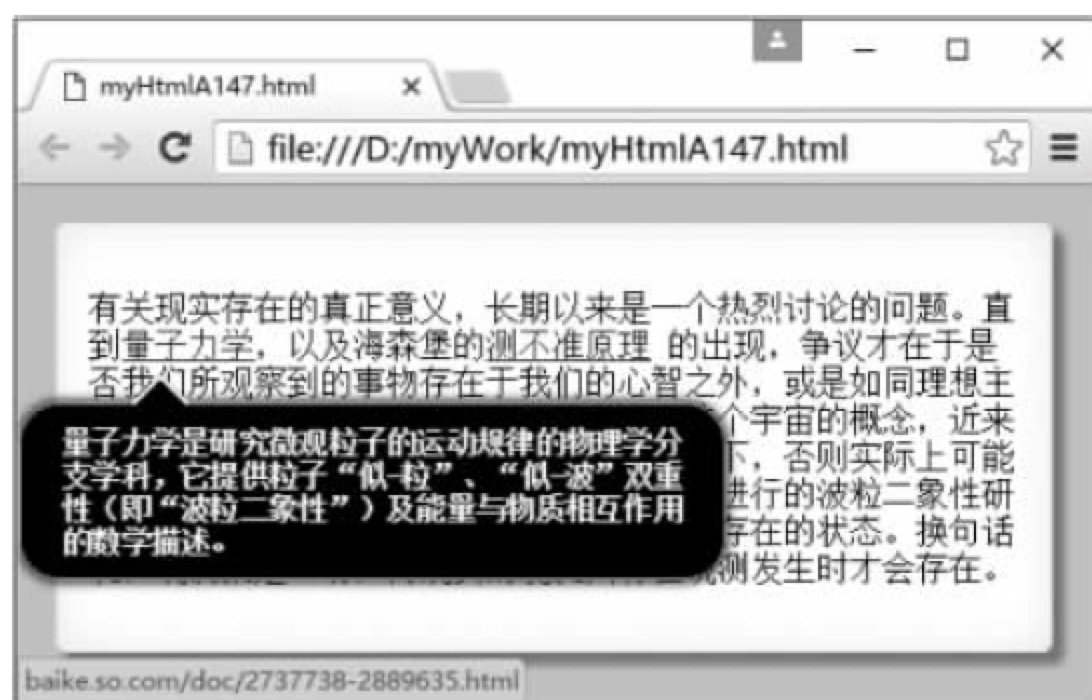


图 446-1

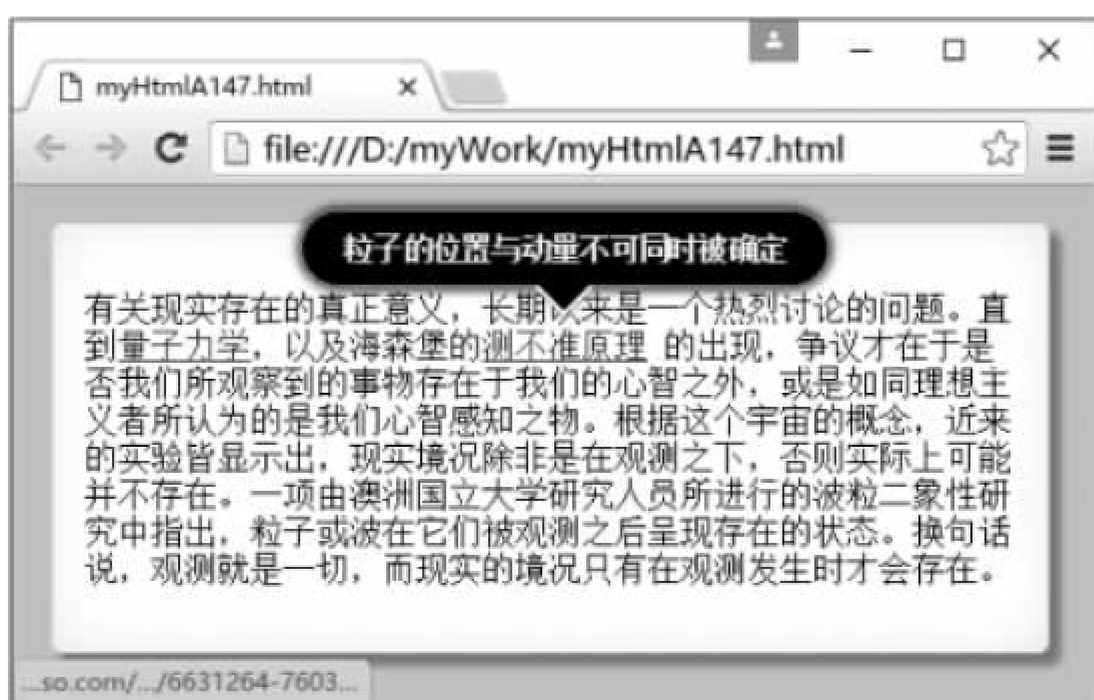


图　446-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(document).tooltip({ })用于创建一个工具提示框，但是自定义样式必须自己设置，即设置.ui-tooltip 和.arrow 类的样式。需要说明的是，tooltip()方法是工具提示框部件(Tooltip Widget)的方法，因此在使用 tooltip()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA147.html。

447　使用阴影滤镜创建带指示箭头的提示框

此实例主要通过设置元素的 filter 属性值为 drop-shadow 创建带阴影特效的箭头提示框。当在 Google Chrome 浏览器中显示该页面时，带阴影特效的箭头提示框如图 447-1 所示。有关此实例的主要代码如下。



图　447-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
```

```
    .box {margin: 40px; padding: 50px;background-color: lightseagreen; position: relative; width:350px;
         height:200px; border-radius: 10px; -webkit-filter: drop-shadow(5px 5px 10px black);}
    .cor {position: absolute; left: -40px; width: 0; height: 0; overflow: hidden; border: 20px solid
         transparent; border-right-color: lightseagreen;}
    h3 { text-align: center;}
    p{text-indent: 30px;}
  </style></head>
<body><div class = "box"><i class = "cor"></i><h3>磁器口古镇</h3><p>磁器口古镇始建于宋代,作为嘉陵江边重要的水陆码头,曾经"白日里千人拱手,入夜后万盏明灯"繁盛一时,被赞誉为"小重庆"。1998 年磁器口古镇被国务院确定为重庆市重点保护传统街,2010 年入选中国历史文化名街,是历经千年变迁而保存至今的重庆市重点保护传统街。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-filter: drop- shadow(5px 5px 10px black)用于为元素(提示框)设置阴影效果,drop-shadow 滤镜添加的阴影效果就好像太阳照在提示框上使其产生影子一样。该滤镜为提示框产生一个快照,使其颜色单一化并将之模糊,然后将模糊结果进行偏离,使其看起来就像提示框的一个阴影。在样式属性中使用多个可选参数,分别用于指定阴影的水平方向偏移距离,垂直方向偏移距离、阴影的模糊半径和阴影的颜色,即 filter: drop-shadow(x 偏移量, y 偏移量,模糊半径,颜色值)。

filter 属性支持的滤镜有 grayscale(灰度)、saturate(饱和度)、hue-rotate(色相旋转)、sepia(复古)、invert(反色)、opacity(透明度)、brightness(亮度)、contrast(对比度)、blur(模糊)、drop-shadow(阴影)。

此实例的源文件名是 myHtmlB109.html。

448 通过图形拼接的方式创建气泡式提示框

此实例主要通过创建一个三角形和一个圆角矩形实现以图形拼接的方式创建气泡式提示框。当在 Google Chrome 浏览器中显示该页面时,带尖角指示的气泡提示框的显示效果如图 448-1 所示。有关此实例的主要代码如下。



图 448-1

```
<!doctype html><html><head><meta charset = "UTF-8">
  <style type = "text/css">
    .myBox{ border-radius: 10px; width: 150px; height:200px; background-image: url(img/B005A.jpg);box
         -shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset; }
    /*三角形*/
```

```
.triangle { position: absolute; top: 60px; left: 160px; width: 0;height: 0;
            border - right: 50px solid #99FFFF; border - bottom: 25px solid transparent;}
/* 圆角矩形 */
.radius - rect { position: absolute; top: 40px; left: 185px; width: 250px; height: 115px; padding:
                 10px; border - radius: 10px; background - color: #99FFFF; }
</style></head>
<body><div class = "myBox"><div class = "triangle"></div><div class = "radius - rect">中央电视台(简称
央视,英语 China Central Television,简称 CCTV),是中华人民共和国国家电视台。1958 年 5 月 1 日试播,9 月 2
日正式播出。初名北京电视台,1978 年 5 月 1 日更名为中央电视台。</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,position: absolute 用于绝对定位,以便三角形和圆角矩形拼接在一起。三角形则主要是通过 border(下边框线和右边框线)透明叠加产生的,如果需要改变三角形的大小,则应该修改 border-right: 50px solid #99FFFF 和 border-bottom: 25px solid transparent 这两个属性值,width: 0 和 height: 0 应该始终为 0。

此实例的源文件名是 myHtmlB200.html。

449　使用透明的线性渐变创建切角提示框

此实例主要通过使用 linear-gradient()方法创建线性渐变背景进行切角(即透明),从而实现提示框的 4 个角被切掉的特殊效果。当在 Google Chrome 浏览器中显示该页面时,被切角的提示框的显示效果如图 449-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .myTips{ position: absolute; top: 50px; left: 50%; transform: translate( - 50%, - 50%); width:
           396px; padding: 10px;text - align: left; font - size: 14px; }
  .myTips{ /* 10px 即 4 个渐变长度,按照左上、右上、右下、左下的顺序切角 */
           /* 切角按照旋转角度并控制渐变长度,即渐变停止位置 */
           background: linear - gradient(135deg, transparent 10px, cyan 0) top left, linear - gradient
( - 135deg, transparent 10px, cyan 0) top right, linear - gradient( - 45deg, transparent 10px, cyan 0)
bottom right, linear - gradient(45deg, transparent 10px, cyan 0) bottom left; background - size: 50%
50%;background - repeat: no - repeat; }
  /* 在矩形的下面添加三角形(即指示箭头) */
  .myTips:after { content: ''; position: absolute; bottom: 0;left: 50%;border: 20px solid transparent;
                  border - top - color:cyan;border - bottom: 0; border - left: 0; margin: 0 0 - 20px - 10px; }
  .myBox{ margin: 105px auto 10px; width: 410px; height: 250px; box - shadow: 2px 2px 8px black; }
  img{ width: 410px; height: 250px;}
 </style></head>
<body><div class = "myTips"> 纽约是一座世界级城市,直接影响着全球的经济、金融、媒体、政治、教育、娱乐与
时尚界,其中联合国总部也位于该市,因此纽约也被公认为世界之都。</div><div class = "myBox"><img src
 = "img/B266A.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,linear-gradient(135deg, transparent 10px, cyan 0) top left 表示以 135°的方向,以 transparent(透明)渐变 10px,其余用 cyan 填充,即创建提示框左上角的切角。当上述代码改为 linear-gradient (135deg,red 10px, lightgray 0px)　top left 时,运行效果如图 449-2 所示,仔细对比两个提示框左上角的变化,大家就可明白渐变透明是怎样切角的。整个切角的提示框就是通过类似的 4 个部分拼接而成的。

此实例的源文件名是 myHtmlB270.html。



图 449-1



图 449-2

450 使用透明的径向渐变创建内圆角提示框

此实例主要通过使用 radial-gradient()方法创建径向渐变背景进行内圆角(即透明),从而实现提示框的 4 个角被切掉的特殊效果。当在 Google Chrome 浏览器中显示该页面时,被内圆角切掉的提示框的显示效果如图 450-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  .myTips { position: absolute; top: 50px; left: 50%; transform: translate( -50%, -50%); width:
          396px; padding: 10px; text-align: left; font-size: 14px; }
  .myTips { /* 10px 即渐变长度,按照左上、右上、右下、左下的顺序实现内圆角 */
          /* 内圆角实际上是将径向渐变的圆心设在 4 个顶点上,透明色自动产生切掉的错觉 */
          background:radial-gradient(circle at top left, transparent 10px, yellowgreen 0) top left,
          radial-gradient(circle at top right, transparent 10px, yellowgreen 0) top right, radial-
          gradient(circle at bottom right, transparent 10px, yellowgreen 0) bottom right, radial-
          gradient(circle at bottom left, transparent 10px, yellowgreen 0) bottom left; background-
          size: 50% 50%; background-repeat: no-repeat; }
  /* 在矩形的下面添加三角形(即指示箭头) */
  .myTips:after { content: ''; position: absolute; bottom: 0; left: 50%;border: 20px solid transparent;
               border-top-color: yellowgreen; border-bottom: 0; border-left: 0; margin: 0 0
               -20px -10px; }
  .myBox { margin: 115px auto 10px; width: 415px; height: 250px;
         border-radius: 10px; box-shadow: 2px 2px 8px black; }
  img { width: 415px; height: 250px; border-radius: 10px;}
 </style></head>
<body><div class = "myTips">东京是日本国的政治、经济、文化中心,海陆空交通枢纽,根据建成区面积、人口以及国民生产总值等指标,东京是世界第一大城市,全球最大的经济中心之一。在 2014 全球城市综合实力排名中位于世界第四。</div>
<div class = "myBox"><img src = "img/B266E.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，radial-gradient(circle at top left，transparent 10px，yellowgreen 0) top left 表示根据 4 个顶点将提示框划分为 4 个等份，然后以左上角为圆心，并用透明色创建一个半径为 10px 的圆，这样此圆与提示框左上角重叠的部分就被透明掉，其他部分则用绿色填充。在 CSS3 中，radial-gradient()方法的主要作用是在颜色与颜色之间产生径向渐变，此实例将渐变半径设置为 0，因此没有颜色在径向上渐变的效果，如果将以上代码改为 radial-gradient(circle at top left，transparent 10px，yellowgreen 150px) top left，则会产生径向的颜色渐变效果，如图 450-2 所示。

此实例的源文件名是 myHtmlB271.html。



图　450-1



图　450-2

451　通过叠加的方式创建箭头风格的提示框

此实例主要通过合理地设置 border-color、width、height 等属性值创建两个大小不同的三角形，用其叠加之后露出的部分作为箭头线条，从而创建箭头风格的提示框。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在图像上，则将在下面出现一个带箭头的提示框，如图 451-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
  $(function() {
   $("img").hover(function(){                    //在鼠标指针悬浮于图像上时显示提示框
    $(".myTooltip").css("display","block");
   },function(){                                 //在鼠标指针离开图像时隐藏提示框
    $(".myTooltip").css("display","none");
  });});
 </script>
 <style type = "text/css">
  /*设置提示框的基本样式*/
```

```
   .myTooltip { display: none; width: 400px; border: 1px solid black; border-radius: 5px;background-
             color: cyan;position: relative; text-align: center; left:calc(50% - 200px); top:25px;
             font-size: 14px;text-align: left; -webkit-transform:rotate(180deg); }
   .myTooltip p{ -webkit-transform:rotate(180deg); }/* 箭头向上、向下与此相关 */
   /* 设置黑色和青色三角形的基本样式,用于裁剪箭头 */
   .myTooltip span { width: 0; height: 0; overflow: hidden; position: absolute; }
   /* 设置黑色三角形的样式 */
   .myTooltip span.myInnerTriangle { border-width: 20px; border-style: solid dashed dashed; border-
                              color: black transparent transparent; left: 80px; bottom: -40px; }
   /* 设置青色三角形的样式 */
   /* 青色和黑色三角形叠加在一起,露出的黑色部分即为箭头的线条宽度 */
   .myTooltip span.myOuterTriangle {border-width: 20px; border-style: solid dashed dashed; border-
                              color: cyan transparent transparent; left: 80px; bottom: -39px; }
   /* 设置图像的基本样式 */
   img{ width:400px;height:200px;border-radius:10px;
      box-shadow: 2px 2px 8px black;}
  </style></head>
 <body><div align="center"><img src="img/B373.jpg"></div>
 <div class="myTooltip"><span class="myInnerTriangle"></span>
    <span class="myOuterTriangle"></span><p>卢浮宫始建于1204年,原是法国的王宫,居住过50位法国
 国王和王后,是法国文艺复兴时期最珍贵的建筑物之一,以收藏丰富的古典绘画和雕刻而闻名于世。</p></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,border-color:black transparent transparent 用于创建一个黑色的三角形,如图 451-2 所示;border-color:cyan transparent transparent 用于创建一个青色的三角形,因为青色三角形的 bottom 为-39px,黑色三角形的 bottom 为-40px,当它们叠加在一起的时候就会露出黑色的边缘,并与青色提示框的黑色边框线连接在一起,起到箭头的指示作用。在 CSS3 中,border-color 属性用于设置 4 个边框线的颜色,此属性可设置一个元素的所有边框中可见部分的颜色,或者为 4 个边框线分别设置不同的颜色。

此实例的源文件名是 myHtmlB373.html。



图 451-1



图 451-2

452　使用 attr()方法将属性值传递给提示框

此实例主要通过使用选择器的 content 属性的可读取 attr()特性实现将自定义属性值传递给提示框的效果。当在 Google Chrome 浏览器中显示该页面时将显示一幅图像，如图 452-1 所示。如果将鼠标指针悬浮在图像上，则将在图像的顶部显示内容来自自定义属性 myMsg 的提示框，如图 452-2 所示。有关此实例的主要代码如下。



图　452-1



图　452-2

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
   .myTips{ position: absolute; top: 60%;left: 50%;
           transform: translate( -50%, -50%);}
   /* 创建提示框 */
   .myTips:hover::before{content:attr(myMsg); position:absolute;top: -68px; left:1px; display:inline
                        -block; background-color: cyan; border-radius: 5px; font-size: 14px;
                        width:388px;text-align:left; cursor:pointer; padding: 5px;}
   /* 创建提示框下面的三角形箭头 */
   .myTips:hover::after{ content:""; border-top:10px solid cyan; border-left:10px solid transparent;
                        border-right: 10px solid transparent; position:absolute; top: -10px; left:
                        40%; }
   img{ width:400px; height:250px;border-radius: 10px; }
  </style></head>
<body><div class = "myTips" myMsg = "故宫博物院建立于 1925 年，是在明朝、清朝两代皇宫及其收藏的基础上
建立起来的中国综合性博物馆，也是中国最大的古代文化艺术博物馆。"><span><img src = "img/B301.jpg">
</span></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content 属性用于插入生成内容，content 属性与 before 选择器及 after 选择器配合使用，将生成内容放在一个元素内容的前面或后面。content 属性的语法格式如下。

```
content:normal|string|attr()|uri()|counter()
```

其中，normal 是默认值；string 表示插入文本内容；attr()表示插入元素的属性值，当使用 attr()获取标签属性名的时候千万不要添加引号；uri()表示插入一个外部资源(图像、声音、视频或浏览器支持的其他任何资源)；counter()表示计数器，用于插入排序标识。

此实例的源文件名是 myHtmlB301.html。

453　通过可选数据列表在文本框中插入内容

此实例主要通过在< input >标签中指定 list 属性实现通过可选数据列表在文本框中插入内容。当在浏览器中显示该页面时，如果在“所学专业：”文本框中输入“计算机”，则将滑出一个自动匹配的可选列表，如图 453-1 所示，选择其中的某一选项，则该选项将插入文本框中。有关此实例的主要代码如下。

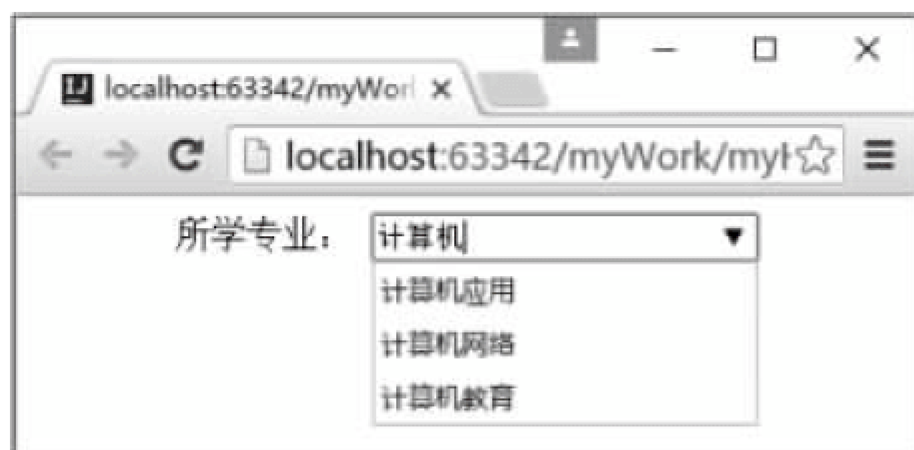


图　453-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><section align = "center">所学专业：<input id = "myMajor" list = "myData"/>
 <datalist id = "myData">
  <option value = "计算机应用"><option value = "计算机网络">
  <option value = "计算机教育"><option value = "视觉传达设计">
  <option value = "信息与计算科学"></datalist></section></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< input >标签的 list 属性指定即将引用的数据列表标签< datalist >的 id，< datalist >标签中的< option >标签定义可选数据的列表选项。需要说明的是，list 属性是< input >标签在 HTML5 中的新属性。

此实例的源文件名是 myHtmlA001.html。

454　使用数据源实现文本框的自动完成功能

此实例主要使用 jquery-ui.min.js 和 jquery-ui.min.css 的 autocomplete()方法实现通过数据源自动完成文本框的输入功能。当在 Google Chrome 浏览器中显示该页面时，如果在文本框中输入字母，例如“J”，则将显示数据源中所有包含字母 J 的选项供用户选择，如图 454-1 所示。如果选择“JavaScript”，则该选项将自动填充到文本框中，如图 454-2 所示。当然，用户也可以使用其他字母进行测试。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <script src = "js/jquery - ui.min.js"></script><script type = "text/javascript">
  $(function() {//通过数组对象设置数据源的数据
```

```
    var mySource = ["ActionScript", "AppleScript", "ASP", "BASIC", "C", "C++", "Fortran", "Java",
"JavaScript", "Perl", "PHP", "Python", "Ruby"];
    //在 source 属性中将数据作为参数传入,供用户进行选择
    $("#myAuto").autocomplete({source: mySource}); });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css">
</head>
<body>
<div align="center">
   <label for="myAuto">请输入您最喜爱的编程语言: </label>
   <input id="myAuto"></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,autocomplete()方法用于根据用户的输入值进行搜索和过滤,让用户快速找到并从预设值列表中选择,该方法中的 source 属性用于指定数据源。例如在此实例中,当在文本框中输入 ja 时则提供 Java 或 JavaScript 供用户选择。需要说明的是,autocomplete()方法是自动完成部件(Autocomplete Widget)的方法,因此在使用 autocomplete()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA162. html。

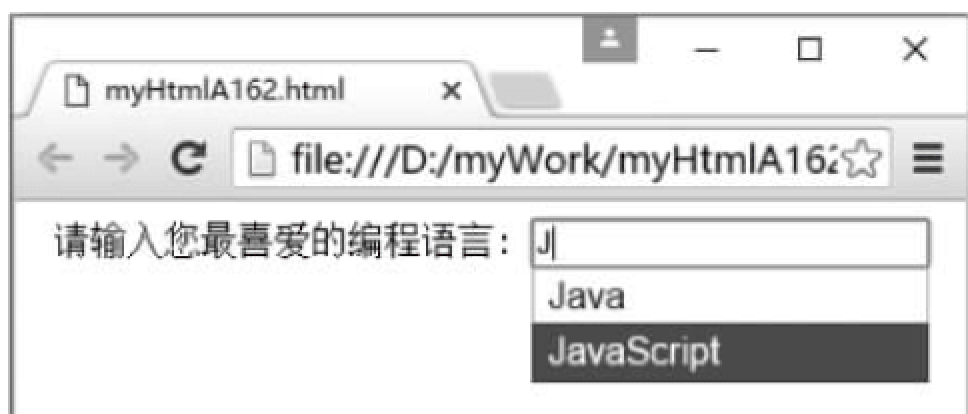


图　454-1



图　454-2

455　为自动完成文本框的下拉列表框添加滚动条

此实例主要使用 jquery-ui. min. js 和 jquery-ui. min. css 的 autocomplete()方法,并定制. ui-autocomplete 类的 CSS 样式,从而实现为自动完成文本框的下拉列表框添加滚动条。当在 Google Chrome 浏览器中显示该页面时,如果在文本框中输入字母,例如"a",因为符合字母 a 的选项较多,超过了规定的下拉列表框高度,因此自动出现了垂直滚动条,如图 455-1 所示。如果在其中选择"AppleScript",则该选项将自动填充到文本框中,如图 455-2 所示。当然,用户也可以使用其他字母进行测试。有关此实例的主要代码如下。

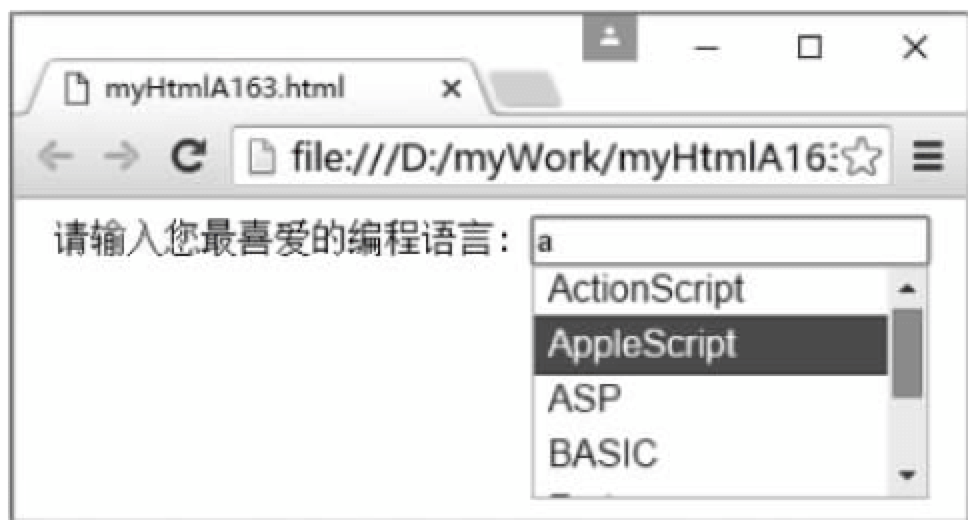


图　455-1



图　455-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {//通过数组对象将下拉候选列表数据存储起来
   var mySource = ["ActionScript", "AppleScript", "ASP", "BASIC", "C", "C++", "Fortran", "Java",
"JavaScript", "Perl", "PHP", "Python", "Ruby"];
   //在 source 属性中将数据作为参数传入,供用户进行选择
   $("#myAuto").autocomplete({source: mySource});  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  /* 设置自动完成文本框的下拉列表框的基本样式 */
  .ui-autocomplete { max-height: 100px; overflow-y: auto; overflow-x: hidden; }
 </style></head>
<body><div align="center"><label for="myAuto">请输入您最喜爱的编程语言: </label><input id=
"myAuto"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,autocomplete()方法用于根据用户的输入值进行搜索和过滤,让用户快速找到并从预设值列表中选择,该方法中的 source 属性用于指定数据源。如果需要对自动完成文本框的下拉列表框进行定制,则应该设置.ui-autocomplete 类的 CSS 样式。需要说明的是,autocomplete()方法是自动完成部件(Autocomplete Widget)的方法,因此在使用 autocomplete()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA163.html。

456 对自动完成文本框的下拉选项进行分类

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 的 autocomplete()方法实现为自动完成文本框的下拉列表框添加分类功能。当在 Google Chrome 浏览器中显示该页面时,如果在文本框中输入字母,例如"p",则将在下拉列表框中分类显示包含字母"p"的选项,如图 456-1 所示。如果在其中选择"pineapple",则该选项将自动填充到文本框中,如图 456-2 所示。当然,用户也可以使用其他字母进行测试。有关此实例的主要代码如下。

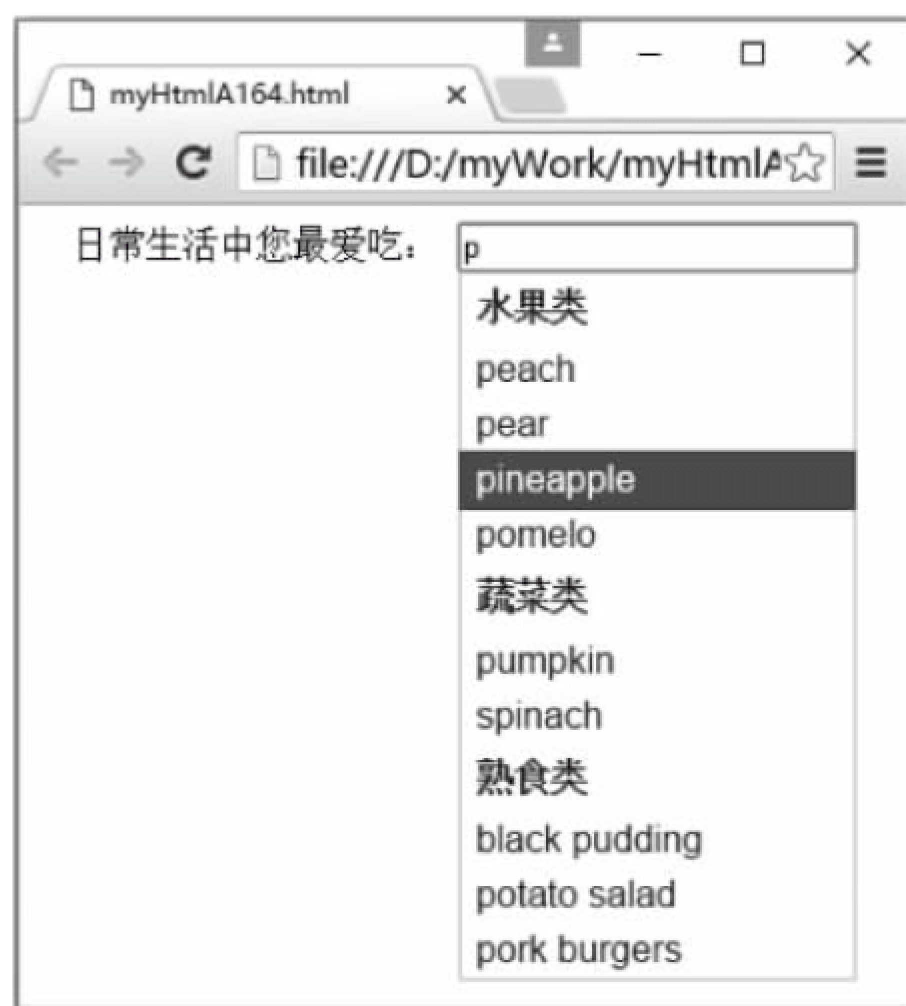


图 456-1



图 456-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  //对自动完成文本框的下拉列表框进行分类
   $.widget( "custom.catcomplete", $.ui.autocomplete, {
    _renderMenu: function( ul, items ) {
     var that = this,currentCategory = "";
     $.each( items, function( index, item ) {
      if ( item.category != currentCategory ) {
       ul.append( "<li class='ui-autocomplete-category'>" + item.category + "</li>" );
       currentCategory = item.category;
      }
      that._renderItemData( ul, item );
   });}});
   $(function() {                              //设置数据源
    var myData = [{ label: "orange", category: "水果类" },{ label: "peach", category: "水果类" },{ label:
"pear", category: "水果类" },{ label: "pineapple", category: "水果类" },{ label: "pomelo", category:
"水果类" },{ label: "pumpkin", category: "蔬菜类" },{ label: "spinach", category: "蔬菜类" },{ label:
"leek", category: "蔬菜类" },{ label: "cucumber", category: "蔬菜类" },{ label: "kale", category: "蔬菜
类" },{ label: "cabbage", category: "蔬菜类" },{ label: "bologna", category: "熟食类" },{ label:
"roast", category: "熟食类" },{ label: "ham", category: "熟食类" },{ label: "black pudding", category:
"熟食类" },{ label: "potato salad", category: "熟食类" },{ label: "pork burgers", category: "熟食类" },
{ label: "salami", category: "熟食类" } ];
    //为文本框添加带分类的自动完成功能
    $( "#myItem" ).catcomplete({ delay: 0,source: myData});  });
 </script>
 <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
  .ui-autocomplete-category { font-weight: bold;padding: 0.2em 0.4em; margin: 0.8em 0 0.2em; line-
height: 1.5;}                                /*设置分类标签的基本样式*/
 </style></head>
<body><div align="center"><label for="myItem">日常生活中您最爱吃：</label>
    <input id="myItem"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，autocomplete()方法用于根据用户的输入值进行搜索和过滤，让用户快速找到并从预设值列表中选择，该方法中的 source 属性用于指定数据源。如果需要对自动完成文本框的下拉列表框的选项进行分类管理，则应该像实例这样重新处理分类选项在下拉列表框中的呈现样式。需要说明的是，autocomplete()方法是自动完成部件(Autocomplete Widget)的方法，因此在使用 autocomplete()方法时必须添加 jquery-ui.min.js 和 jquery-ui.min.css 两个文件。

此实例的源文件名是 myHtmlA164.html。

457　启用或禁用文本框的自动完成功能

此实例主要通过设置<form>标签和<input>标签的 autocomplete 属性实现启用或禁用表单或者文本框的自动完成功能。当在浏览器中显示该页面时，双击“公司名称：”或“公司地址：”文本框，则将向下滑出下拉列表框，显示之前在此文本框中输入的历史记录，选择其中的选项则会自动将其填充到文本框中，如图 457-1 所示。“联系电话：”文本框没有自动完成功能，因此双击此文本框不会滑出下拉列表框。有关此实例的主要代码如下。

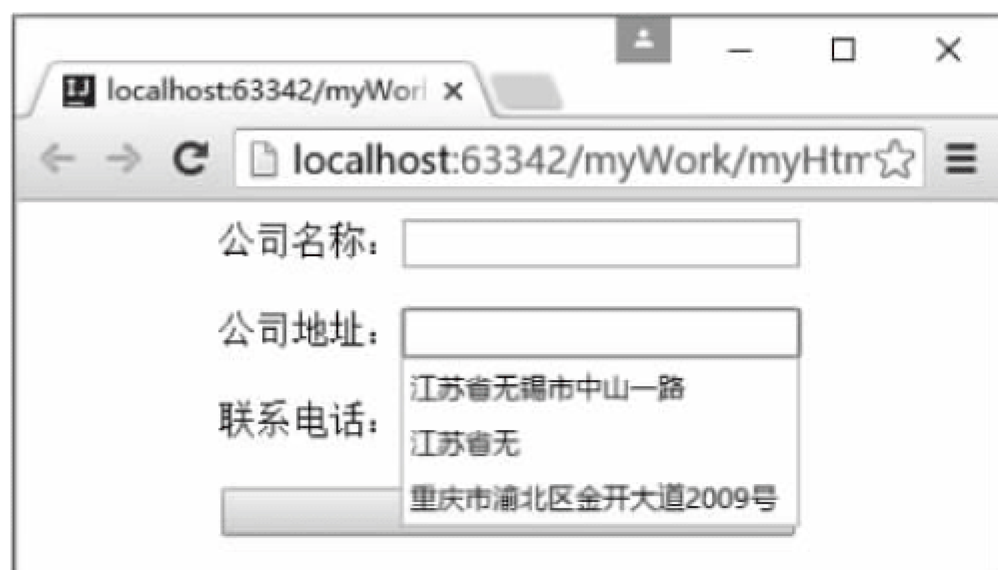


图 457-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><form  autocomplete = "on" align = "center" width = "300px">
    公司名称：<input type = "text" id = "myName"/><br><br>
    公司地址：<input type = "text" id = "myAddress"/><br><br>
    联系电话：<input type = "text" id = "myPhone" autocomplete = "off"/><br><br>
    <button type = "submit" style = "width: 250px">提交登记信息</button></form>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，autocomplete 属性是<form>标签和<input>标签在 HTML5 中的新属性，该属性规定 form 或 input 元素应该拥有自动完成功能。如果设置 form 元素的 autocomplete 属性为 on，则该 form 中的 input 元素的 autocomplete 属性自动为 on，除非明确设置 input 元素的 autocomplete 属性为 off。在此实例中，“公司名称：”和“公司地址：”文本框虽然没有设置 autocomplete 属性，但这两个文本框的 autocomplete 属性自动为 on，因此具有自动完成功能；“联系电话：”文本框重新设置了 autocomplete 属性为 off，因此没有自动完成功能。这种情形通常出现在登录窗口中，“用户名称：”文本框具有自动完成功能，但“用户密码：”通常没有自动完成功能。

注意，autocomplete 适用于<form>标签，以及 text、search、url、telephone、email、password、datepickers、range、color 类型的<input>标签。

此实例的源文件名是 myHtmlA018.html。

458 禁止用户在文本框中粘贴剪切板的内容

此实例主要通过重新处理(拦截)文本框<input>的 paste 事件响应方法实现禁止用户向文本框中粘贴剪切板的内容。当在 Google Chrome 浏览器中显示该页面时，在“沪市账号：”文本框获得焦点时，如果试图向其粘贴内容，则将弹出一个消息框提示“抱歉，已经禁止使用粘贴方式！请手动输入。”，如图 458-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
   $(function(){//在用户对输入框进行粘贴操作时拦截用户操作，使粘贴内容无效
    $("#myAccount").on("paste",function(){
     alert("抱歉，已经禁止使用粘贴方式！请手动输入。");
     return false;  });});
 </script>
 <style type = "text/css">
```

```
div { margin-top: 2px; margin-left: 2px; }
</style></head>
<body><div align="center"><label for="myAccount">沪市账号: </label><input type="text" id=
"myAccount"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,以下内容是文本框的 paste 事件响应方法。

```
$("#myAccount"). on("paste",function(){
  alert("抱歉,已经禁止使用粘贴方式!请手动输入。");
  return false; });
```

注意,如果设置 return true,则仍然能够在弹出消息框之后进行粘贴操作,因此必须设置 return false。根据同样的思路,如果需要禁止用户复制文本框中的内容,则代码可以修改为如下。

```
$("#myAccount").on("copy",function(){
  alert("抱歉,已经禁止使用复制此部分内容!请手动输入。");
  return false;});
```

此实例的源文件名是 myHtmlA238.html。

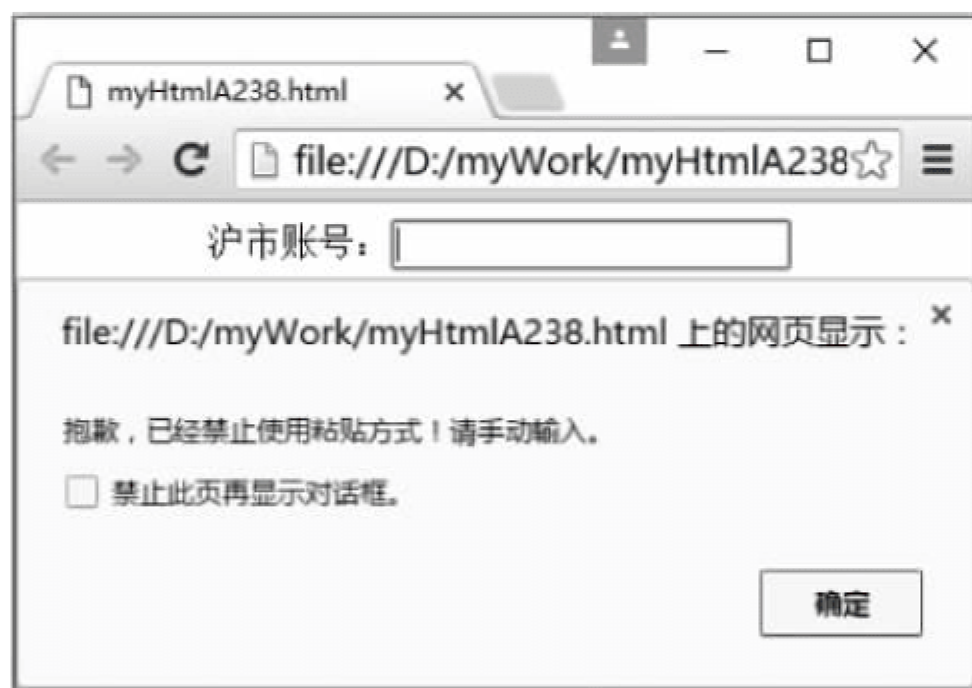


图　458-1

459　允许或禁止对文本框中的单词进行拼写检查

此实例主要通过设置<textarea>标签和<input>标签的 spellcheck 属性实现允许或禁止对用户输入的英文单词进行拼写检查。当在浏览器中显示该页面时,如果在"中文名称:"文本框中输入错误的英文单词,将不会出现错误提示;如果在"英文简历:"文本框中输入错误的英文单词,例如"dreamm",按下空格键后将会在该英文单词下面添加红色的波浪线,如图 459-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body>中文名称:<input type="text" id="myName" spellcheck="false"
                style="width:300px"/><br><br>
      英文简历:<textarea spellcheck="true" rows="5" cols="40"
                id="myBrief"></textarea></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< textarea >标签和< input >标签的 spellcheck 属性是 HTML5 新增的属性，该属性规定是否对元素内容进行拼写检查。当元素的 spellcheck 属性被设置为 true 时将对元素的文本进行拼写检查；当元素的 spellcheck 属性被设置为 false 时则不对元素的文本进行拼写检查。

此实例的源文件名是 myHtmlA019. html。

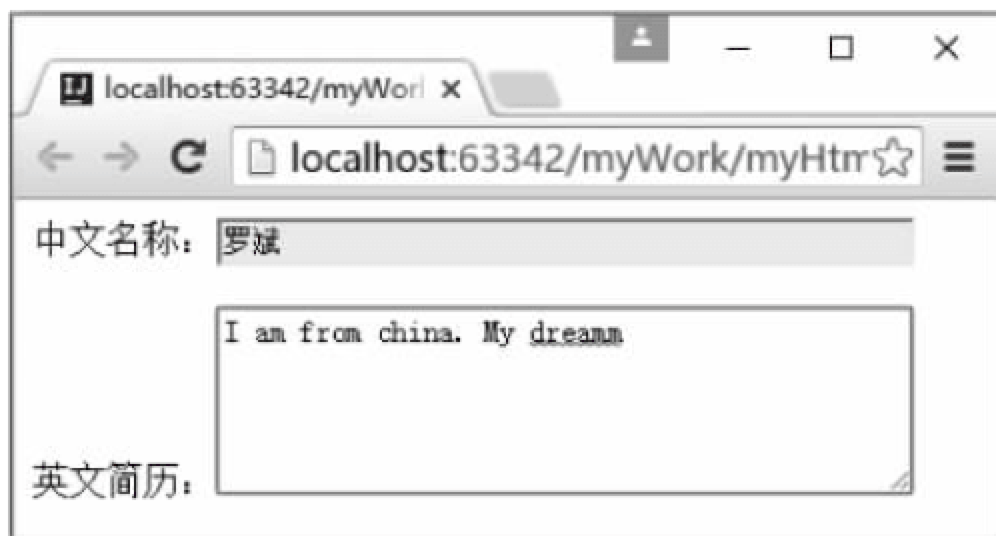


图 459-1

460 设置 pattern 属性规范文本框中数据的输入

此实例主要通过设置 pattern 属性实现规范用户在文本框中的数据输入。当在浏览器中显示该页面时，如果用户在“身份证号码:”文本框中输入的内容不满足 18 位居民身份证号码的格式，则将在该文本框附近滑出一个提示框，提示“请与所请求的格式保持一致。”，如图 460-1 所示。有关此实例的主要代码如下。

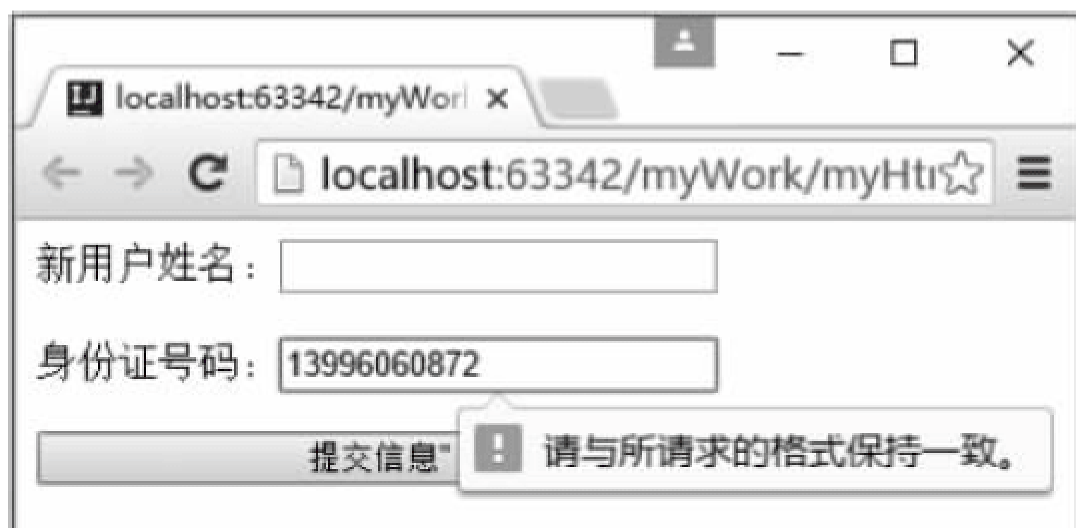


图 460-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><form id = "myForm">
 <label>新用户姓名:<input type = "text" id = "myUser"></label><br><br>
 <label>身份证号码: <input type = "text" pattern = "^\d{18} $ " required placeholder = "请与所请求的格式保持一致。" id = "myID"/></label><br><p><input type = "submit" value = 提交信息" style = "width:270px"/> <p/></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，pattern 属性是 HTML5 新增的属性。pattern 属性规定用于验证输入字段的模式，此模式是正则表达式，例如“^\d{18} $”。pattern 属性适用于 text、search、url、telephone、email、password 等< input >类型。

此实例的源文件名是 myHtmlA025. html。

461　在文本框中设置灰色的输入提示信息

此实例主要通过设置 placeholder 属性实现在文本框中设置默认的灰色输入提示信息。当在浏览器中显示该页面时，“用户名称：”文本框的提示信息是“不支持汉字用户名称”，“用户密码：”文本框的提示信息是“只能输入字母、数字及其组合”，在任意一个文本框中输入字符，则该文本框的提示信息自动消失，如图 461-1 所示。有关此实例的主要代码如下。



图　461-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><form action = "" method = "post" >
    <label id = "myUser">用户名称：<input type = "text"  placeholder = "不支持汉字用户名称"></label>
<br><br>
    <label id = "myPassword">用户密码：<input type = "text" maxlength = "11" placeholder = "只能输入字
母、数字及其组合"> </label></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，placeholder 属性是 HTML5 新增的属性。对 input(text、search、url、telephone、email、password)、textarea 指定 placeholder 属性即会对用户的输入进行提示，提示用户此文本框期待什么样的输入。当文本框获得焦点时提示字符自动消失。

此实例的源文件名是 myHtmlA023.html。

462　在提交表单时自动检查无内容的文本框

此实例主要通过设置 required 属性实现在提交表单时自动检查未输入内容的文本框。当在浏览器中显示该页面时，如果“用户名称：”文本框中未输入内容，单击“登录”按钮，则将在“用户名称：”文本框附近滑出一个提示框，提示“请填写此字段。”，如图 462-1 所示。有关此实例的主要代码如下。

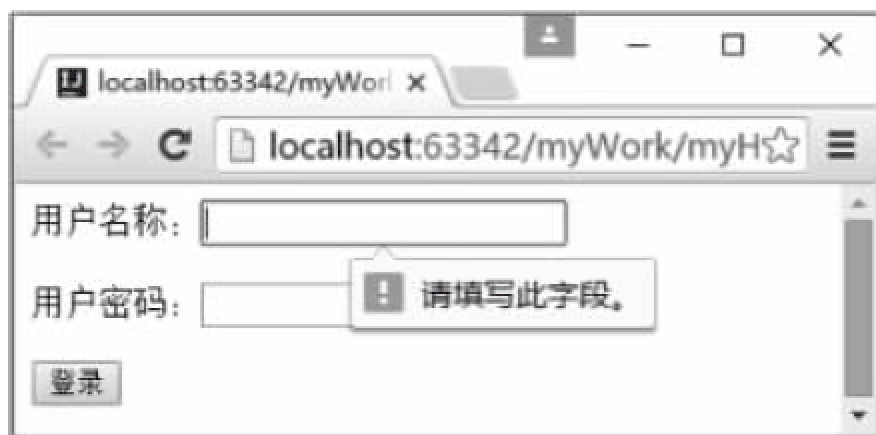


图　462-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><form action = "" method = "post">
```

```
    <label id="myUser">用户名称:<input type="text" required></label><br><br>
    <label id="myPassword">用户密码:<input type="text" required/></label>
    <p><input  type="submit" value="登录"/></p></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，required 属性是 HTML5 新增的属性。对 input(text、search、url、telephone、email、password、date pickers、number、checkbox、radio、file)元素来说，一旦设置了 required 属性，则将在提交表单时自动检查该元素是否为空。

此实例的源文件名是 myHtmlA024.html。

463　在提交表单时校验文本框中的电子邮箱

此实例主要设置<input>标签的 type 属性为 email，从而实现在提交表单时自动校验用户在文本框中输入的电子邮箱格式。当在浏览器中显示该页面时，如果在“电子邮箱:”文本框中输入了错误的邮箱，例如“binluobin#163.com”，则在单击表单的“提交注册信息”按钮后将在出现错误的文本框附近显示错误信息，如图 463-1 所示。有关此实例的主要代码如下。



图　463-1

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><form style="width: 400px;">
    用户名称: <input type="text" name="myName"/><br><br>
    电子邮箱: <input type="email" name="myEmail"/><br><br>
    <button type="submit" value="提交" style="width: 250px">提交注册信息</button>
  </form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，email 属性值是 HTML5 中<input>标签的 type 属性的新值，该属性值指明文本框应该包含 e-mail 地址，在提交表单时将会自动验证该文本框中的内容是否符合 e-mail 的标准格式。

此实例的源文件名是 myHtmlA007.html。

464　在提交表单时校验文本框中的网址格式

此实例主要设置<input>标签的 type 属性为 url，从而实现在提交表单时自动校验用户在文本框中输入的网址格式。当在浏览器中显示该页面时，如果在“公司网站:”文本框中输入了错误的网址，例如“http.//www.163.com”，则在单击表单的“提交注册信息”按钮后将在出现错误的文本框附近显示错误信息，如图 464-1 所示。有关此实例的主要代码如下。



图　464-1

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><form style="width: 400px;">
    公司名称：<input type="text" name="myName"/><br><br>
    公司网站：<input type="url" name="myUrl"/><br><br>
    <button type="submit" value="提交" style="width: 250px">提交注册信息</button>
  </form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，url 属性值是 HTML5 中<input>标签的 type 属性的新值，该属性值指明文本框应该包含 URL 地址的输入字段，在提交表单时将会自动验证该文本框中的内容是否符合 URL 地址的标准格式。

此实例的源文件名是 myHtmlA008.html。

465　在提交表单时校验文本框中的数字范围

此实例主要设置<input>标签的 type 属性为 number，从而实现在提交表单时自动校验用户在文本框中输入的数字是否在预置的有效范围内。当在浏览器中显示该页面时，如果在"年龄："文本框中输入的数字大于上限，则将在"年龄："文本框附近显示如图 465-1 所示的错误信息；如果在"年龄："文本框中输入的数字小于下限，将在"年龄："文本框附近显示如图 465-2 所示的错误信息。有关此实例的主要代码如下。



图　465-1



图　465-2

```
<!doctype html><html><head><meta charset=UTF-8></head>
<body><form style="width: 400px;">
    姓名：<input type="text" name="myName"/><br><br>
    年龄：<input type="number" name="myAge" min="25" max="40"/>岁(25-40之间的数字)<br><br>
<button type="submit" value="提交" style="width: 250px">提交注册信息</button></form></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，number 属性值是 HTML5 中<input>标签的 type 属性的新值，该属性值指明文本框用于包含数字值的输入字段。一旦将<input>标签的

type 属性设置为 number，则应该参考表 465-1 设置< input >标签的相关属性，从而指定文本框中的数字的有效范围和步长值。

表 465-1　设置< input >标签的相关属性

属　　性	值	说　　明
max	数字	规定允许的最大值
min	数字	规定允许的最小值
step	数字	规定合法的数字间隔(如果 step="3"，则合法的数是－3、0、3、6 等)
value	数字	规定默认值

此实例的源文件名是 myHtmlA009.html。

466　在文本框获得焦点时显示发光的边框线

此实例主要在 CSS 样式中设置文本框的 box-shadow 属性，从而实现在文本框获得焦点时显示发光的边框线。当在 Google Chrome 浏览器中显示该页面时，单击任意一个文本框，则将在该文本框的四周出现蓝色的光圈，如图 466-1 所示。有关此实例的主要代码如下。

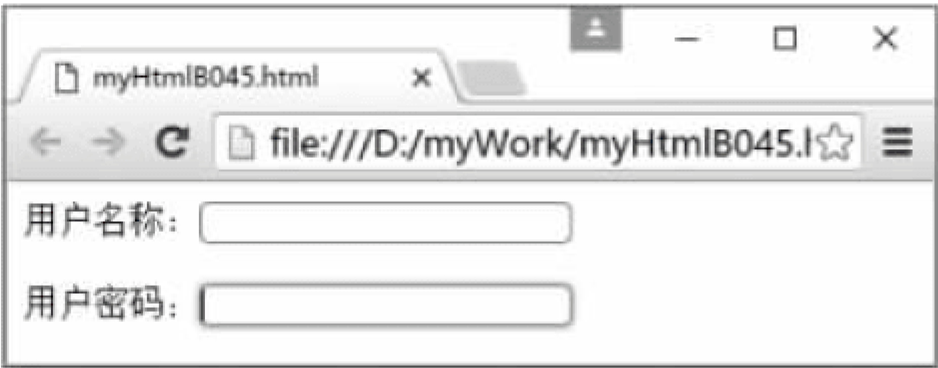


图　466-1

```
<!doctype html><html><head><meta charset = UTF - 8>
<style>
  input {transition: all 0.30s ease - in - out; - webkit - transition: all 0.30s ease - in - out;  - moz -
        transition: all 0.30s ease - in - out; border - radius: 4px; outline: none; border: #35A5E5 1px
        solid;}
  input:focus {box - shadow: 0 0 5px rgba(81, 203, 238, 1);  - webkit - box - shadow: 0 0 5px rgba(81, 203,
             238, 1);  - moz - box - shadow: 0 0 5px rgba(81, 203, 238, 1);}
 </style></head>
<body><form action = "" method = "post"><label id = "myUser">用户名称:<input type = "text"></label>
<br><br><label id = "myPassword">用户密码:<input type = "text"></label></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，box-shadow：0 0 5px rgba(81，203，238，1)用于为文本框设置发光的边框线，为了使其不像是阴影效果而达到发光的效果，这里采用了明亮的蓝色。如果用户使用过 Twitter，可能已经留意到在输入框获得焦点后它的边框会有蓝色发光的效果，并且这里运用了 transition 属性，使得发光效果有很平滑的过渡。

此实例的源文件名是 myHtmlB045.html。

467　通过 autofocus 属性设置当前文本框

此实例主要通过设置 autofocus 属性实现文本框自动获得焦点。当在浏览器中显示该页面时，由于“用户密码:”文本框设置了 autofocus 属性，因此该文本框的光标处于闪烁状态，如图 467-1 所示。

有关此实例的主要代码如下。

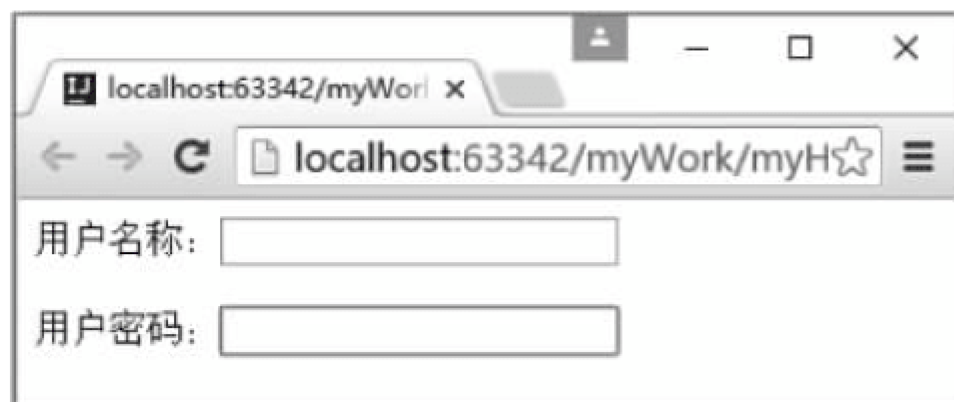


图　467-1

```
<!doctype html><html><head><meta charset = UTF - 8></head>
<body><form action = "" method = "post">
 <label id = "myUser">用户名称:<input type = "text"  maxlength = "20"></label><br><br><label id =
"myPassword">用户密码:<input type = "text"  maxlength = "11" autofocus = "autofocus"></label></form>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，autofocus 属性是 HTML5 新增的属性。对 input(所有类型)、select、textarea 指定 autofocus 属性，则可以实现让元素在页面加载后自动获得焦点。一个页面只能有一个元素具有 autofocus 属性，如果同时设置多个，则第一个生效。这个属性对登录页面很有用，可提升用户体验，有时登录页面仅有用户名称和密码，页面加载后用户需要手动定位到输入框才能输入，有了 autofocus，在页面打开后即可直接输入。

此实例的源文件名是 myHtmlA022.html。

468　通过 label 的 control 属性访问文本框

此实例主要实现通过 label 元素的 control 属性访问 input 元素。当在浏览器中显示该页面时，单击“设置默认值”按钮，则将在“联系电话:”文本框中显示预置的默认值“13996060872”，如图 468-1 所示。有关此实例的主要代码如下。

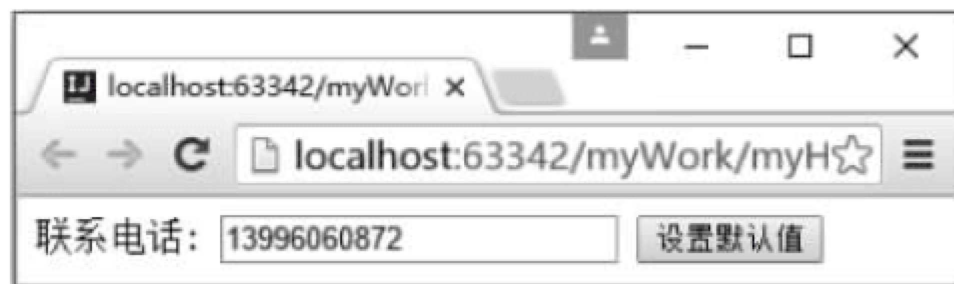


图　468-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script type = "text/javascript">
  function setDefault(){//设置默认值
   //var myLabel = $ ("#myLabel").get(0);
   var myLabel = document.getElementById('myLabel');
   var myText = myLabel.control;
   myText.value = '13996060872'; }
 </script></head>
<body><form action = "" method = "post">
   <label id = "myLabel"> 联系电话: <input type = "text" name = "text" value = "" maxlength = "11"></label>
<input type = "button" value = "设置默认值" onclick = "setDefault()"></form></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,label 元素的 control 属性是 HTML5 新增的属性,通过该属性可以简单地实现访问 label 元素嵌套的子元素。在此实例中 input 元素是 label 元素嵌套的子元素,因此 label 元素的 control 属性对应的即是 input 元素。

此实例的源文件名是 myHtmlA021.html。

469 使段落文本实现类似文本框的编辑功能

此实例主要通过设置< p >标签的 contenteditable 属性使段落文本实现类似于文本框的编辑功能。当在浏览器中显示该页面时,段落文本中的"内容简介:"是不可编辑的,如图 469-1 所示;单击"允许编辑内容简介"按钮,"内容简介:"则处于可编辑状态,因此可以在此段落文本中直接进行编辑,如图 469-2 所示;单击"禁止编辑内容简介"按钮,则"内容简介:"恢复为不可编辑状态。



图 469-1



图 469-2

有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script>
  $(document).ready(function() {
   $('#myBtnEnable').click(function() {        //允许编辑内容简介
    $('#myContent').prop("contenteditable",true);
    $('#myContent').css("background","lightgray");
    $('#myContent').focus();
   });
   $('#myBtnDisable').click(function() {       //禁止编辑内容简介
    $('#myContent').prop("contenteditable",false);
    $('#myContent').css("background","white"); });});
 </script></head>
<body><div style = "width:400px">
   <p id = "myContent">内容简介:«西游记»是一部中国古典神魔小说,为中国"四大名著"之一。书中讲述唐朝法师西天取经的故事,表现了惩恶扬善的古老主题。</p>
   <input class = "input" type = "button"  value = "允许编辑内容简介" id = "myBtnEnable" style = "width:195px"/>
   <input class = "input" type = "button"  value = "禁止编辑内容简介" id = "myBtnDisable" style = "width:195px"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$('#myContent').prop("contenteditable",true)用于设置< p >标签的 contenteditable 属性为 true,即允许编辑< p >标签的内容;$('#myContent').prop("contenteditable",false)用于设置< p >标签的 contenteditable 属性为 false,即禁止编辑< p >标签的内容。在 HTML5 中,contenteditable 属性规定是否可编辑元素的内容,该属性的可用属性值如表 469-1 所示。

表 469-1　contenteditable 属性的可用值

值	说　明
true	规定可以编辑元素内容
false	规定无法编辑元素内容

此实例的源文件名是 myHtmlA002.html。

470　在单个段落中使用省略号代替超长文本

此实例主要通过设置<p>标签的 text-overflow 属性为 ellipsis 实现在单个段落中使用省略号代替超长文本。当在 Google Chrome 浏览器中显示该页面时，第一个和倒数第二个段落由于内容超长，因此在末尾自动显示省略号，如图 470-1 所示。有关此实例的主要代码如下。



图　470-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   p { text-overflow: ellipsis; white-space: nowrap; width: 350px; overflow: hidden;
       padding: 8px; margin: 10px; background: lightblue; border-radius: 5px; font-weight: bold; }
 </style></head>
<body><p>架构即未来：现代企业可扩展的 Web 架构、流程和组织(原书第 2 版)</p>
  <p>JavaScript DOM 编程艺术</p><p>Python 编程 从入门到实践</p>
  <p>深入分析 Java Web 技术内幕</p>
  <p>数据结构与算法分析：Java 语言描述(原书第 3 版)</p>
  <p>用户体验要素：以用户为中心的产品设计</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-overflow 属性规定当文本溢出包含元素时应如何处置，该属性的语法格式如下。

```
text-overflow: clip|ellipsis|string
```

其中，clip 表示直接剪掉超长的文本；ellipsis 表示使用省略号来代替超长部分的文本；string 表示使用给定的字符串来代替超长部分的文本。

此实例的源文件名是 myHtmlB126.html。

471　在可滚动段落底部添加渐变透明遮罩层

此实例主要通过设置<div>元素的-webkit-mask-image属性实现在滚动段落的底部添加渐变透明的遮罩层。当在Google Chrome浏览器中显示该页面时，无论是否滑动滚动条，在段落的底部始终有一个渐变透明的遮罩层，如图471-1所示。有关此实例的主要代码如下。



图　471-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myBox { display: inline-block;  width: 200px; height: 218px; padding: 15px; margin: 20px 100px 20px
100px; background-color: #FFF; border: 1px solid #EEE; overflow: auto; position: relative; border-
radius: 2px; border: black solid 1px; -webkit-mask-image: linear-gradient(to bottom, rgba(0, 0, 0,
1) 70%, rgba(0, 0, 0, 0) 100%); }
  body { background: lightblue; }
 </style></head>
<body><div class="myBox">天宝元年(公元742年),由于玉真公主和贺知章的交口称赞,玄宗看了李白的诗
赋,对其十分仰慕,便召李白进宫。李白进宫朝见那天,玄宗降辇步迎,"以七宝床赐食于前,亲手调羹"。玄宗问
到一些当世事务,李白凭半生饱学及长期对社会的观察,胸有成竹,对答如流。玄宗大为赞赏,随即令李白供奉
翰林,职务是给皇上写诗文娱乐,陪侍皇帝左右。玄宗每有宴请或郊游,必命李白侍从,使用他敏捷的诗才,赋诗
纪实。虽非记功,也将其文字流传后世,以盛况向后人夸示。李白受到玄宗如此的宠信,同僚不胜艳羡,但也有
人因此而产生了嫉恨之心。</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-mask-image：linear-gradient(to bottom，rgba(0，0，0，1) 70%，rgba(0，0，0，0) 100%)表示在div元素的垂直方向的70%处开始至100%处结束创建一个渐变透明的遮罩层。在CSS中，linear-gradient()方法用于以线性渐变的方式创建图像，如果想创建以对角线方式渐变的图像，可以使用to top left这样的多关键字方式来实现，默认的渐变方向只能绘制一个最简单的线性渐变。linear-gradient()方法的语法格式如下。

```
<linear-gradient> = linear-gradient([ [ <angle> | to <side-or-corner> ] ,]? <color-stop>
[, <color-stop>]+)
<side-or-corner> = [left | right] || [top | bottom]
<color-stop> = <color> [ <length> | <percentage> ]?
```

其中，<angle>表示用角度值指定渐变的方向(或角度)，取值为 to left 表示设置渐变为从右到左，相当于 270deg；取值为 to right 表示设置渐变从左到右，相当于 90deg；取值为 to top 表示设置渐变从下到上，相当于 0deg；取值为 to bottom 表示设置渐变从上到下，相当于 180deg。这是默认值，等同于留空不写。

<color-stop>用于指定渐变的起止颜色。<color>指定颜色。<length>用长度值指定起止色的位置，不允许为负值。<percentage>用百分比指定起止色的位置。下面是 linear-gradient()方法常用的几种渐变形式。

```
linear-gradient(#FFF, #333);
linear-gradient(to bottom, #FFF, #333);
linear-gradient(to top, #333, #FFF);
linear-gradient(180deg, #FFF, #333);
linear-gradient(to bottom, #FFF 0%, #333 100%);
```

此实例的源文件名是 myHtmlB208.html。

472　使用 textarea 的拖曳标志实现 CSS 交互

此实例主要通过使用 textarea 的拖曳标志实现纯 CSS 交互，即图片的滑进与滑出。当在 Google Chrome 浏览器中显示该页面时，拖动右侧的红色方框即有一张图片滑进、滑出，如图 472-1 所示。有关此实例的主要代码如下。

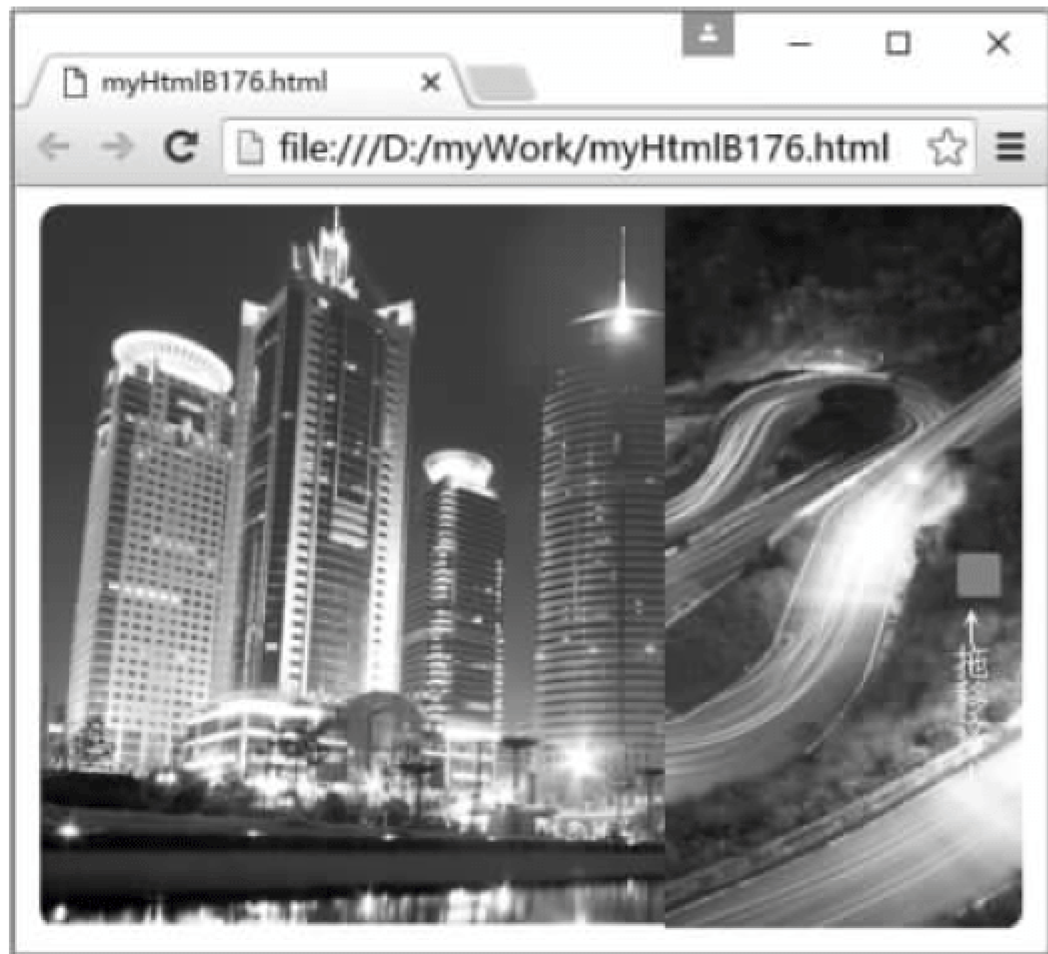


图　472-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .box { width:450px; position:relative; margin-left: auto; margin-right: auto; overflow: hidden;
border-radius: 10px;}
  .cover { height: 100%;  position: absolute; top: 0; right: -20px;
           background:url(img/B099A.jpg) no-repeat center; }
  .cover textarea { width: 20px; height: 20px; margin-top: 160px; border: 0px; padding: 0; opacity:0;
cursor: e-resize; position:relative; z-index:1; left: -20px;max-width:495px; }
  /*红色方框代替拖曳标志*/
```

```
  .instead { width: 20px; height: 20px;  background-color: red; position: absolute; right:10px; top:
160px; border-radius: 2px; }
  .instead::after { content: '↑拖移这个'; line-height: 1; margin-top: 26px;
                right:5px; color: #FFF; font-style: normal; position:absolute;}
 </style></head>
<body><center><div class="box"><img src="img/B099B.jpg" />
   <div class="cover"><textarea></textarea></div>
   <i class="instead"></i></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，max-width:495px 表示最大滑动宽度，以防止拖曳时越界，否则图片在向左拖曳时，如果图片的右边界拖出盒子的左边框，则无法再拖回图片。在此实例中拖曳功能看似与 textarea 元素无关，但实际上使用了 textarea 元素的 resize 功能，如果设置 resize:none，则拖曳功能消失。

此实例的源文件名是 myHtmlB176.html。

473　获取在 textarea 中使用鼠标选择的文本

此实例主要通过 window.getSelection()方法实现获取用户在 textarea 中使用鼠标选择的文本内容。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标在 textarea 中选择“世界”，松开鼠标则将弹出一个消息框显示选择的文本，如图 473-1 所示。有关此实例的主要代码如下。



图　473-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $('#myBrief').select(function() {//获取选中的文本内容
     var myObj = window.getSelection();
     var myText = myObj.toString();
     alert('选中的文本为: '+ myText); });});
 </script></head>
<body><textarea rows="10" cols="55" id="myBrief">1945年，第二次世界大战接近尾声，作为邪恶轴心重
要成员的日本，其嚣张态势已成强弩之末。是年，决定战局走向的冲绳岛战役拉开序幕，成千上万斗志昂扬的美
国大兵被派往冲绳，等待他们的则是敌军重兵防守、凶险异常的钢锯岭。在这群人中间，却有一个不愿拿起武器
的军医。他名叫戴斯蒙德·道斯(安德鲁·加菲尔德 Andrew Garfield 饰)，来自美国的弗吉尼亚。太平洋战争爆
发之际，瘦弱的戴斯蒙德志愿成为救死扶伤的军医而应征入伍。可因童年和家庭的原因，他始终不愿拿起枪支操
```

```
练,为此宁愿背上拒服兵役的罪名被送上军事法庭。几经周折,戴斯蒙德最终和战友来到了钢锯岭。枪林弹雨,转瞬之间无数人应声倒地。在信仰和信念的支持下,戴斯蒙德仅凭一己之力拯救了数十条濒死的生命……本片根据真人真事改编。</textarea>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,<textarea>标签是定义多行的文本输入控件,在文本区中可容纳无限数量的文本,用户可以通过 cols 和 rows 属性来规定 textarea 的尺寸。使用 window.getSelection()方法则可以获取用户在<textarea>中使用鼠标选择的文本内容,该方法的返回值即是选中的文本。

此实例的源文件名是 myHtmlA113.html。

474　通过禁止选择文本来实现禁止复制内容

此实例主要通过设置-webkit-user-select 属性为 none 来禁止用户选择文本,从而实现禁止用户复制网页内容。当在 Google Chrome 浏览器中显示该页面时,上面一行"下面这段演示文本禁止复制粘贴"是可以进行复制粘贴的,下面的演示文本则根本无法使用鼠标和键盘进行选择,因此也无法实现复制粘贴,如图 474-1 所示。有关此实例的主要代码如下。



图　474-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  p { margin: 5px;  width: 395px; padding: 10px; border: 1px solid #BFBFBF; background-color: lightseagreen; box-shadow: 2px 2px 3px gray; -webkit-user-select: none; }
  div { width: 395px; text-align: center;}
 </style></head>
<body><div>下面这段演示文本禁止复制粘贴</div><p>春节,是农历正月初一,又叫阴历年,俗称"过年"。这是我国民间最隆重、最热闹的一个传统节日。春节的历史很悠久,它起源于殷商时期年头岁尾的祭神祭祖活动。按照我国农历,正月初一古称元日、元辰、元正、元朔、元旦等,俗称年初一,到了民国时期,改用公历,公历的一月一日称为元旦,把农历的一月一日叫春节。</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-user-select: none 用于禁止用户选择文本。在 CSS3 中,user-select 属性用于设置或检索是否允许用户选中文本,该属性的语法格式如下。

```
user-select: none | text | all | element
```

其中,各属性值的意义如下。

（1）none：该属性值表示文本不能被选择。

（2）text：该属性值表示可以选择文本，此为默认值。

（3）all：该属性值表示当所有内容作为一个整体时可以被选择。如果双击或者在上下文上单击子元素，那么被选择的部分将是以该子元素向上回溯的最高祖先元素。

（4）element：该属性值表示可以选择文本，但选择范围受元素边界的约束。

此实例的源文件名是 myHtmlB197.html。

475　通过设置 oncopy()的返回值禁止复制内容

此实例主要设置页面 body 的 oncopy()方法的返回值为 false，从而实现禁止用户将选择的页面内容复制到剪贴板。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标在页面中选择部分文字，然后再右击选择的部分文字，在弹出的快捷菜单中选择"复制(C)"命令，则将弹出一个消息框，提示"此页面内容禁止复制！"，如图 475-1 所示。当然，在其他文本编辑器(例如记事本)中无法粘贴此部分文字。有关此实例的主要代码如下。



图　475-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  p { margin: 5px;   width: 395px; padding: 10px; border: 1px solid # BFBFBF; background - color:
lightseagreen;box - shadow: 2px 2px 3px gray;   }
 </style></head>
<body  oncopy = "alert('此页面内容禁止复制!'); return false;"><p>下面这段演示文本禁止复制粘贴：春节，是农历正月初一，又叫阴历年，俗称"过年"。这是我国民间最隆重、最热闹的一个传统节日。春节的历史很悠久，它起源于殷商时期年头岁尾的祭神祭祖活动。按照我国农历，正月初一古称元日、元辰、元正、元朔、元旦等，俗称年初一，到了民国时期，改用公历，公历的一月一日称为元旦，把农历的一月一日叫春节。</p>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< body oncopy="alert('此页面内容禁止复制!')；return false;">表示禁止复制选择的页面文字内容，如果将 oncopy()方法的返回值设置为 true，例如< body　oncopy=" return true;">，则可以正常复制粘贴选择的部分文字。

此实例的源文件名是 myHtmlB314.html。

476　允许或禁止超长的数字或者单词跨行显示

此实例主要通过设置单元格的 word-break 属性实现允许或禁止超长的数字或者单词在表格的单元格中跨行显示。当在 Google Chrome 浏览器中显示该页面时，表格的第 2 列第 2 行由于允许一个英文单词或超长数字跨行显示，因此 had、several、Building、Province 等单词和数字 20000000000000 都在两行中显示；表格的第 2 列第 3 行由于禁止一个英文单词或超长数字跨行显示（这也是默认值），因此没有超长数字和单词跨行显示；如果在单元格中的一个数字或单词不能在一行中显示，则会另起新行显示，如图 476-1 所示。有关此实例的主要代码如下。

图　476-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .table { margin: 5px 0 25px;border: 1px solid #CCC; border-collapse: collapse;}
  .table td, .table th {border: 1px solid #DDD; padding: 5px 10px;}
  .table th {background-color: #F0F0F0;}
  .table_fixed {table-layout: fixed;}
  /* 使用省略号代替单行超长文本 */
  .cell { white-space: nowrap; overflow: hidden; text-overflow: ellipsis;}
  /* 允许单词或大数字跨行显示 */
  .breakLine {word-break: break-all;}
 </style></head>
<body>
 <table class="table table_fixed" width="100%" cellspacing="0" cellpadding="0">
 <tr><th width="30%">特性</th><th width="50%">演示</th></tr>
 <tr><td>在单行中使用省略号代替超长内容</td>
      <td><div class="cell">Responsibilities included word processing, data entry, filing, and sometimes bookkeeping</div></td></tr>
 <tr><td>允许一个英文单词或超长数字跨行显示</td>
      <td><div class="breakLine">Be interested in research and survey, had ever organized a team and gone to several villages to survey about the Building a new socialist countryside 20000000000000 in Wuming County, Guangxi Province</div>
      </td></tr>
 <tr><td>禁止一个英文单词或超长数字跨行显示</td>
```

```
    <td><div>Be interested in research and survey, had ever organized a team and gone to several villages to survey about the Building a new socialist countryside 20000000000000 in Wuming County, Guangxi Province</div></td></tr></table></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-overflow 属性规定当文本溢出包含元素时应如何处理，如果该属性值为 ellipsis，则使用省略号来代表超长部分的文本。word-break 属性规定自动换行的处理方法，该属性的语法格式如下。

```
word-break: normal|break-all|keep-all
```

其中，属性值 normal 表示使用浏览器默认的换行规则；属性值 break-all 表示允许在单词内换行；属性值 keep-all 表示只能在半角空格或连字符处换行。

此实例的源文件名是 myHtmlB142.html。

477 设置元素中所有单词的首字母是否大写

此实例主要通过设置元素的 text-transform 属性控制元素中的所有单词的首字母是否大写。当在 Google Chrome 浏览器中显示该页面时，单击“首字母大写”按钮，则下面显示的每个英文单词的首字母均大写，如图 477-1 所示；单击“全部大写”按钮，则下面显示的每个英文单词的所有字母均大写，如图 477-2 所示；单击“全部小写”按钮，则下面显示的每个英文单词的所有字母均小写，如图 477-3 所示。有关此实例的主要代码如下。



图 477-1

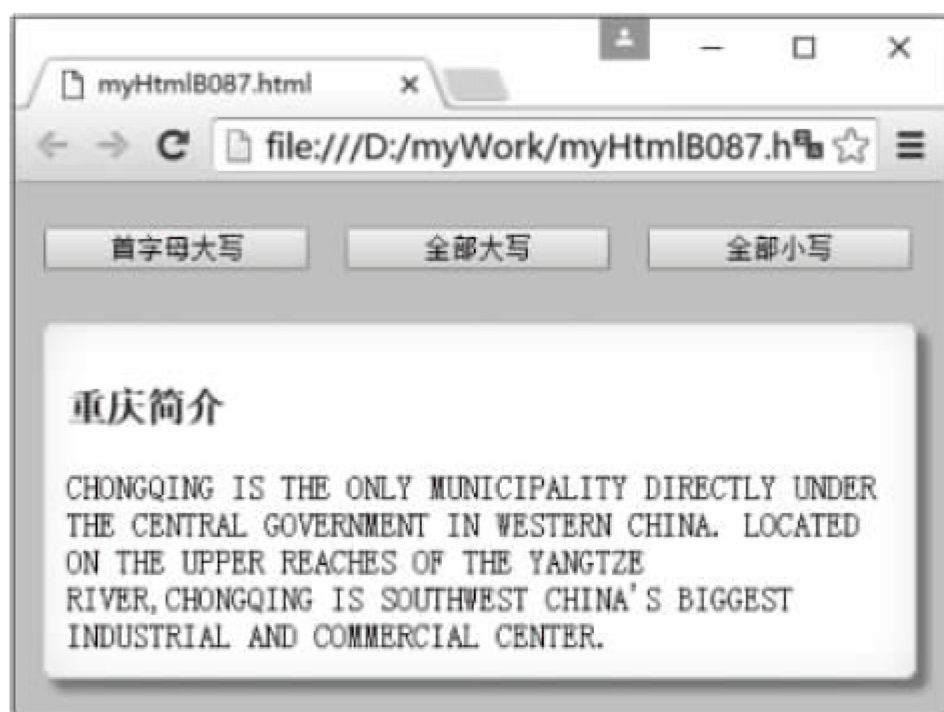


图 477-2



图 477-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnCapitalize").click(function() {       //首字母大写
    $("div").css("text-transform", "capitalize");
   });
   $("#myBtnUppercase").click(function() {        //全部大写
    $("div").css("text-transform", "uppercase");
   });
   $("#myBtnLowercase").click(function() {        //全部小写
    $("div").css("text-transform", "lowercase");  });});
 </script>
 <style>
  div { display: inline-block; width: 400px; height: 150px;margin: 5px; background-color: #FFF;
border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset; position:
relative; border-radius: 5px; padding: 10px; text-transform:none; }
  body {background-color: lightgray;}
  input{width:128px; margin:5px;}
 </style></head>
<body><p><input type="button" value="首字母大写" id="myBtnCapitalize"/>
    <input type="button" value="全部大写" id="myBtnUppercase"/>
    <input type="button" value="全部小写" id="myBtnLowercase"/></p>
<div><h3>重庆简介</h3>Chongqing is the only municipality directly under the central government in
western China. Located on the upper reaches of the Yangtze River,Chongqing is southwest China's biggest
industrial and commercial center.</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("div").css("text- transform","capitalize")用于设置 div 块中的所有单词的首字母均大写。在 CSS3 中，text-transform 属性用于检索或设置对象中文本的大小写，text-transform 属性的语法格式如下。

```
text-transform: none | capitalize | uppercase | lowercase | full-width
```

其中，none 表示无转换；capitalize 表示将每个单词的第一个字母转换成大写；uppercase 表示将每个单词转换成大写；lowercase 表示将每个单词转换成小写；full-width 表示将所有字符转换成 fullwidth 形式，如果字符没有相应的 fullwidth 形式，将保留原样，这个值通常用于排版拉丁字符和数字等表意符号。

此实例的源文件名是 myHtmlB087.html。

478　设置段落中的部分文字是否带下画线

此实例主要设置元素的 text-decoration 属性为 underline(即下画线)，并设置子元素的 display 属性为 inline-block，从而实现段落中的部分文字有下画线、部分文字没有下画线。当在 Google Chrome 浏览器中显示该页面时，“乐山大佛”标题带下画线，内容没有下画线(但它们都在一个 p 元素内)；“大足石刻”标题和内容均有下画线(它们也都在一个 p 元素内)，如图 478-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div {display: inline-block; width: 400px;height: 270px;margin: 10px; background-color: #FFF;
border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; position:
relative;border-radius: 5px;}
```

```
  p { padding-left: 15px; padding-right: 10px;font-size: 16px; font-family: 华文楷体;color:black;
text-decoration: underline;}
  p span {display: inline-block;}
  body { background-color: lightgray;}
 </style></head>
<body><div><p>乐山大佛<span>又名凌云大佛,位于四川省乐山市南岷江东岸凌云寺侧,濒大渡河、青衣江和
岷江三江汇流处。大佛为弥勒佛坐像,通高 71 米,是中国最大的一尊摩崖石刻造像。</span></p>
    <p>大足石刻位于中国重庆大足区境内,距重庆主城九区 167 公里,有 74 处 5 万余尊宗教石刻造像。它是
唐末、宋初时期的宗教摩崖石刻,以佛教题材为主,儒、道教造像并陈,尤以北山摩崖造像和宝顶山摩崖造像为
著。北山摩崖造像长约三百多米,是世界文化遗产。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,p span {display: inline-block;}用于将对象呈现效果控制在一行内,注意实例在制造这种效果时是另起一行;text-decoration: underline 用于为元素添加下画线。在 CSS3 中,text-decoration 属性规定添加到文本的修饰,修饰的颜色由 color 属性设置。这个属性允许对文本设置某种效果,例如添加下画线。如果后代元素没有自己的装饰,在祖先元素上设置的装饰会延伸到后代元素中。text-decoration 属性的可选值包括 none(默认值,定义标准的文本)、underline(定义文本下的一条线)、overline(定义文本上的一条线)、line-through(定义穿过文本的一条线)、blink(定义闪烁的文本)、inherit(规定应该从父元素继承 text-decoration 属性值)。

此实例的源文件名是 myHtmlB086.html。

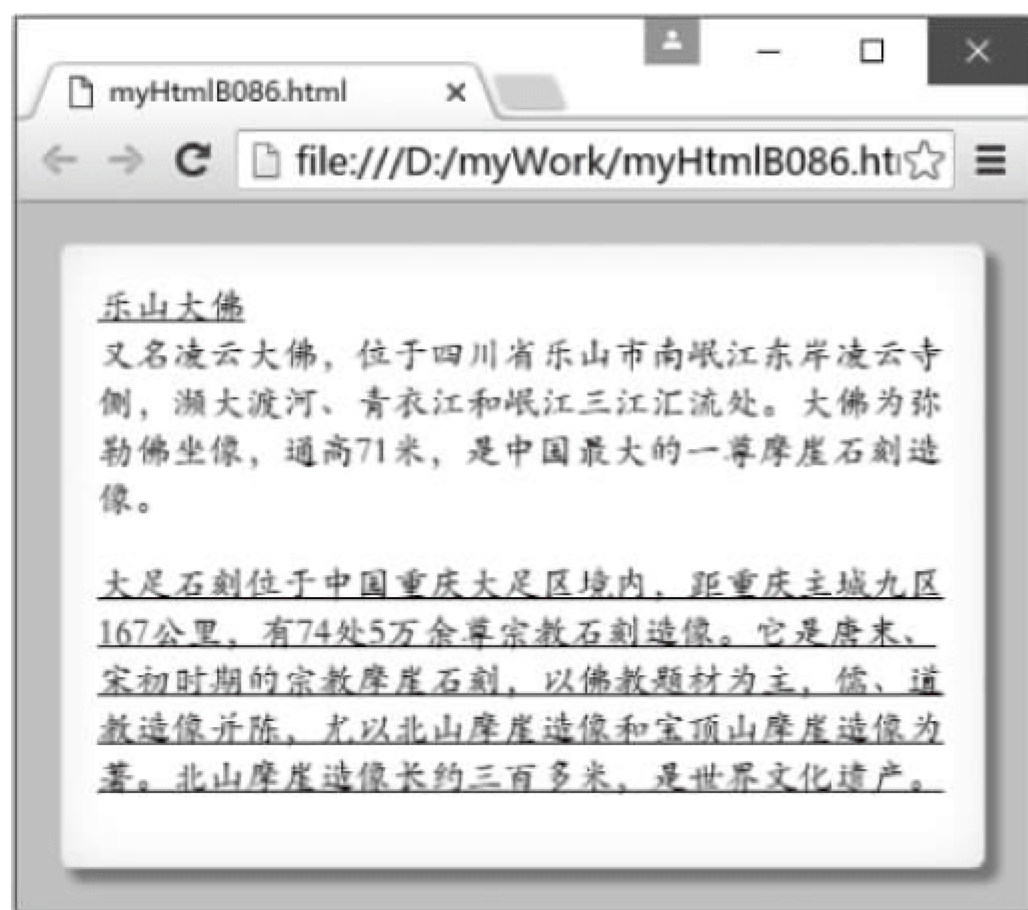


图 478-1

479 对段落中的部分文字添加自定义的下画线

此实例主要通过定制 span 元素的 border-bottom-color、border-bottom-style、border-bottom-width 等属性实现对段落中的部分文字添加自定义的下画线。当在 Google Chrome 浏览器中显示该页面时,段落中的所有名字下面均有一条红色的下画线,如图 479-1 所示。有关此实例的主要代码如下。

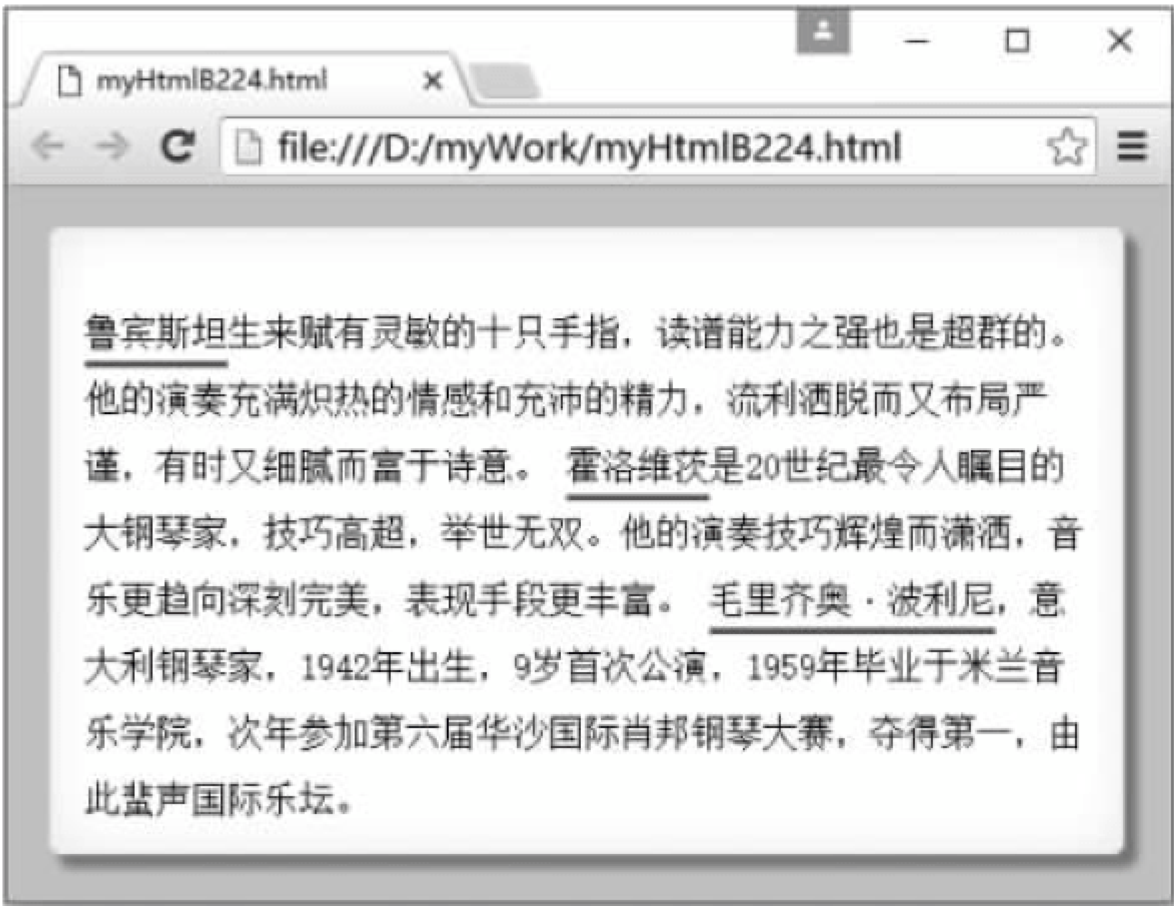


图　479-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   /* 绘制阴影框 */
   div { display: inline-block;width: 450px;height: 250px; padding: 15px; margin: 10px; background-color: #FFF;border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; position: relative; border-radius: 5px;}
   body { background-color: lightgray;}
   /* 取消段落默认的下画线属性 */
   p{ line-height: 30px; text-decoration: none; }
   /* 定制下画线的风格 */
   span { border-bottom-color: red; border-bottom-style:solid; border-bottom-width:3px; padding-bottom: 5px; }
 </style></head>
<body><div><p><span>鲁宾斯坦</span>生来赋有灵敏的十只手指,读谱能力之强也是超群的。他的演奏充满炽热的情感和充沛的精力,流利洒脱而又布局严谨,有时又细腻而富于诗意。<span>霍洛维茨</span>是20世纪最令人瞩目的大钢琴家,技巧高超,举世无双。他的演奏技巧辉煌而潇洒,音乐更趋向深刻完美,表现手段更丰富。<span>毛里齐奥·波利尼</span>,意大利钢琴家,1942年出生,9岁首次公演,1959年毕业于米兰音乐学院,次年参加第六届华沙国际肖邦钢琴大赛,夺得第一,由此蜚声国际乐坛。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,< span >元素被用来组合文档中的行内元素;border-bottom-color 属性用于设置元素的下边框颜色,该属性只能定义纯色,而且只有当边框的样式是一个非 none 或 hidden 的值时边框才可能出现;border- bottom-width 属性用于设置元素的下边框的宽度,只有当边框样式不是 none 时才起作用,如果边框样式是 none,边框宽度实际上会重置为 0;border-bottom-style 属性用于设置元素的下边框样式,该属性支持的样式的意义如表 479-1 所示。

表 479-1　border-bottom-style 属性支持的样式

属　性　值	描　　述
none	定义无边框
hidden	与 none 相同,不过应用于表时除外,对于表,hidden 用于解决边框冲突
dotted	定义点状边框,在大多数浏览器中呈现为实线
dashed	定义虚线,在大多数浏览器中呈现为实线

续表

属　性　值	描　　述
solid	定义实线
double	定义双线，双线的宽度等于 border-width 的值
groove	定义 3D 凹槽边框，其效果取决于 border-color 的值
ridge	定义 3D 垄状边框，其效果取决于 border-color 的值
inset	定义 3D inset 边框，其效果取决于 border-color 的值
outset	定义 3D outset 边框，其效果取决于 border-color 的值
inherit	规定应该从父元素继承边框样式

此实例的源文件名是 myHtmlB224.html。

480　创建背景符号对文本自定义下画线

此实例主要通过线性渐变创建符号（缩小之后就成为一个点）来设置背景，从而使文字的底部显示类似于下画线的重复符号（点）。当在 Google Chrome 浏览器中显示该页面时，“欧阳修”“李清照”“司马相如”“王昌龄”等的底部分别显示有省略号风格的下画线，如图 480-1 所示。有关此实例的主要代码如下。

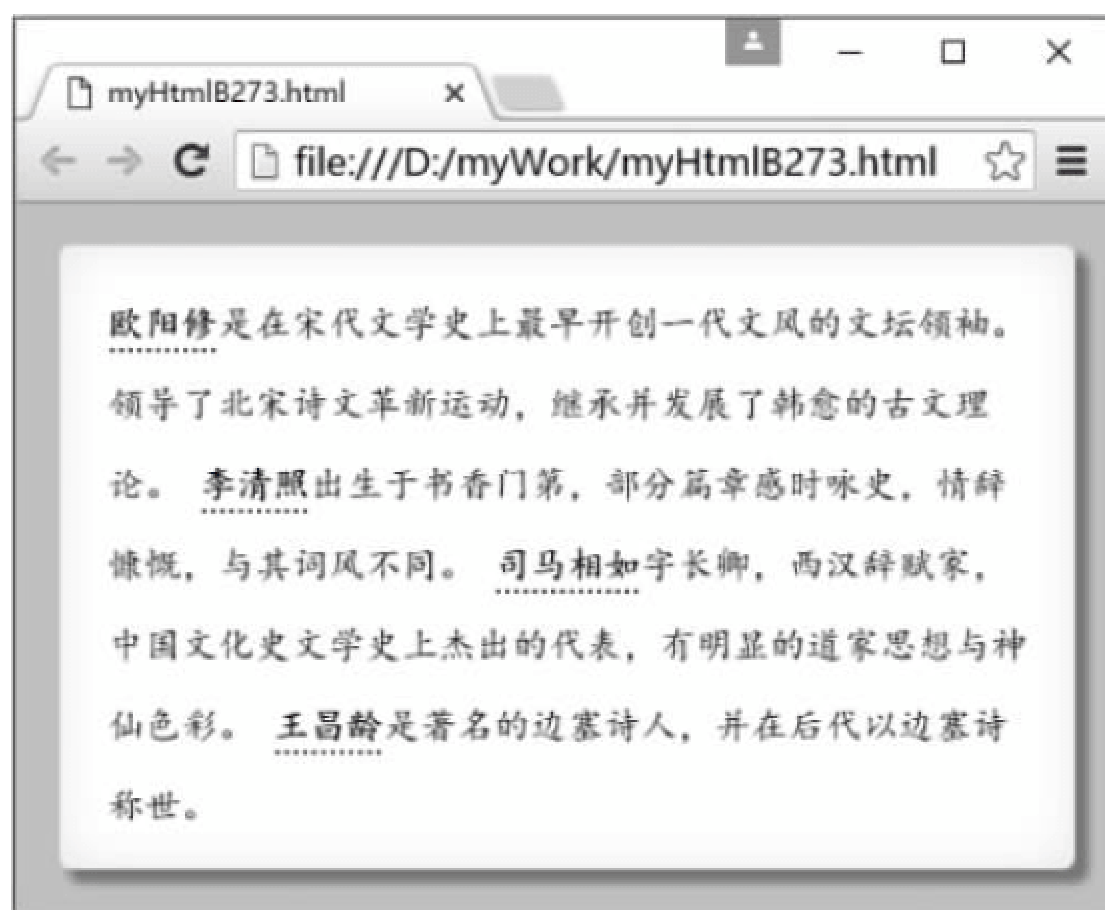


图　480-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /*绘制阴影框*/
  div { display: inline-block; width: 400px; height: 250px; padding-left: 20px; padding-right: 20px;
padding-bottom: 20px; margin: 10px; background-color: #FFF; border: 1px solid #EEE; box-shadow: 5px
5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; position: relative; border-radius: 5px; }
  body { background-color: lightgray; }
  p{line-height: 35px; text-decoration: none; font-family: 楷体;font-size: 16px; }
  /*定制下画线的风格,生成背景图像*/
  span { position: absolute; height: 35px; font-size: 16px; background: linear-gradient(-45deg,
transparent 40%, deeppink 0, deeppink 60%, transparent 0) 0 1em,linear-gradient(45deg, transparent
40%, deeppink 0, deeppink 60%, transparent 0) 0.1em 1em;  background-size: 0.3em 0.15em; background
-repeat: repeat-x; background-position: 0 28px; }
```

```
</style></head>
<body><div><p><span>欧阳修</span>欧阳修是在宋代文学史上最早开创一代文风的文坛领袖。领导了北宋诗文革新运动,继承并发展了韩愈的古文理论。<span>李清照</span>李清照出生于书香门第,部分篇章感时咏史,情辞慷慨,与其词风不同。<span>司马相如</span>司马相如字长卿,西汉辞赋家,中国文化史文学史上杰出的代表,有明显的道家思想与神仙色彩。<span>王昌龄</span>王昌龄是著名的边塞诗人,并在后代以边塞诗称世。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background-size 属性用于规定背景图像(符号)的尺寸,该属性的第一个值用于设置宽度,第二个值用于设置高度。background-size：0.3em 0.15em 中的 em 实际上是垂直测量,一个 em 等于任何字体中的字母所需要的垂直空间,而和它占据的水平空间没有任何关系,如果字体大小是 16px,那么 1em=16px,因此原代码完全等价于 background-size：4.8px 2.4px。background- position 属性用于设置背景图像位置,必须先指定 background-image 属性。该属性通常提供两个参数值,第一个用于横坐标,第二个用于纵坐标,如果只提供一个,该值将用于横坐标,纵坐标默认为 50%。

此实例的源文件名是 myHtmlB273.html。

481　使用自定义的波浪线设置文本的下画线

此实例主要通过使用 linear-gradient()方法创建渐变图形,从而实现使用自定义的波浪线设置文本的下画线。当在 Google Chrome 浏览器中显示该页面时,使用自定义的红色波浪线设置的文本下画线效果如图 481-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  p { margin: 1px; display: inline - block; width: 400px; height: 210px; font - size: 18px;color: black; padding: 15px 20px; box - shadow: 0 5px 5px darkgrey; border - radius: 10px;background - color: #EEE;line - height: 40px; }
  .myWave { position: absolute; background: linear - gradient(45deg, transparent, transparent 45%, red, transparent 55%, transparent 100%), linear - gradient( - 45deg, transparent, transparent 45%, red, transparent 55%, transparent 100%); background - size: 10px 10px;  background - repeat: repeat - x; height: 33px; background - repeat: repeat - x; background - position: 0 28px;}
 </style></head>
<body><p>最佳故事片: <span class = "myWave">«巴山夜雨»、«天云山传奇»</span><br>
  最佳科教片: <span class = "myWave">«生命与蛋白质 - 人工合成胰岛素»</span><br>
  最佳纪录片: <span class = "myWave">«蛇口奏鸣曲»</span><br>
  最佳儿童片: <span class = "myWave">«烛光里的微笑»</span><br>
  最佳合拍故事片: <span class = "myWave">«宋氏三姐妹»</span><br></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,background：linear- gradient(45deg,transparent,transparent 45%,red,transparent 55%,transparent 100%),linear-gradient(-45deg,transparent,transparent 45%,red,transparent 55%,transparent 100%)用于生成渐变图形,如图 481-2 所示。此图形并不是纯粹的红色波浪线,而是通过设置 height：33px 和 background-position：0 28px 隐藏此渐变图形的下面部分,这样才使文本的下画线出现类似于波浪线的效果。在 CSS3 中,text-decoration-style：wavy 用于设置装饰线的风格为波浪线,但是目前除了火狐浏览器(Mozilla Firefox)外,其他浏览器目前都不支持 text-decoration-style 属性。

此实例的源文件名是 myHtmlB332.html。



图 481-1

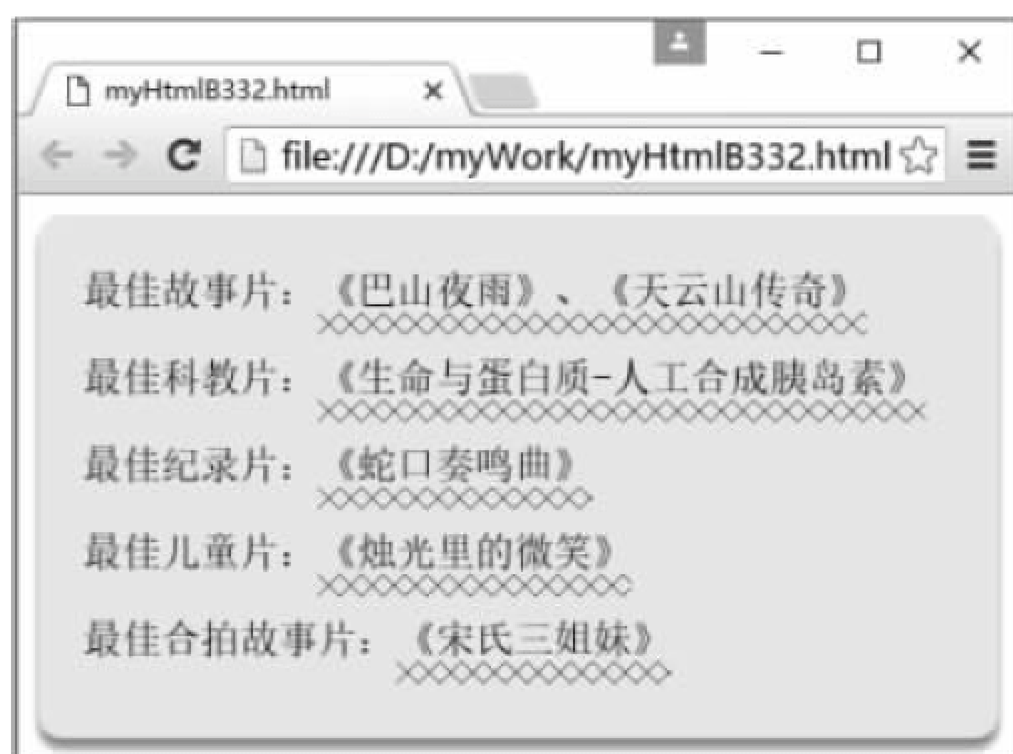


图 481-2

482 通过 background-image 创建下画线

此实例主要通过设置 background-image 属性值为 linear-gradient 实现使用虚线装饰超链接的下画线。当在 Google Chrome 浏览器中显示该页面时将以红色的虚线显示超链接(第二个)的悬浮状态，如图 482-1 所示。有关此实例的主要代码如下。

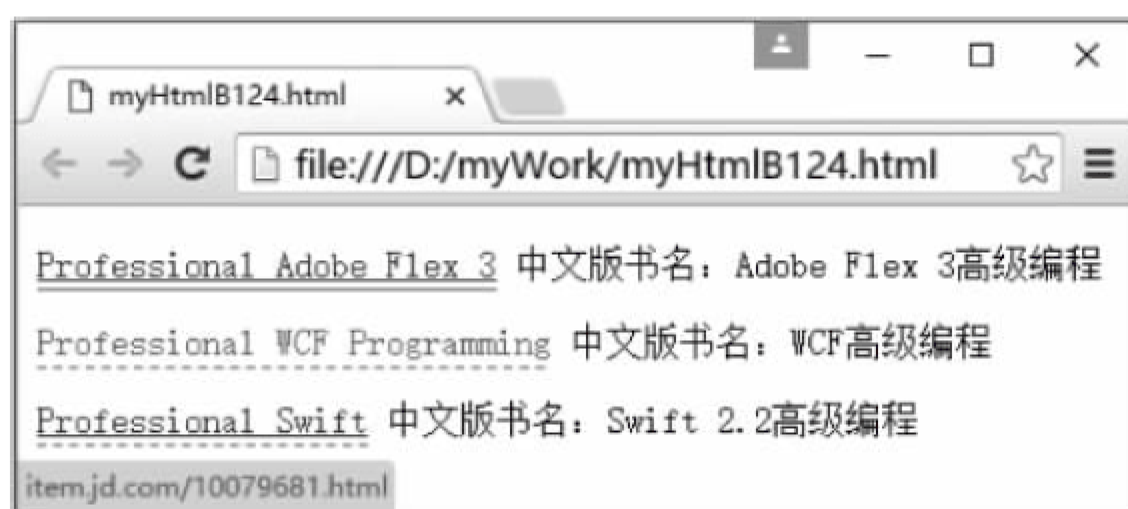


图 482-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .solid {padding-bottom: 4px;
      background-image: linear-gradient(to top, red, green 1px, transparent 1px);}
  .dashed { padding-bottom: 4px; background: linear-gradient(to right, red, green 4px, transparent
4px) repeat-x 0 bottom/7px 1px; }
  .dashed:hover,.solid:hover{color: red; text-decoration: none;}
  .solid:hover {background-image: linear-gradient(to top, red, green 1px, transparent 1px);}
  .dashed:hover { background-image: linear-gradient(to right, red, green 4px, transparent 4px);}
 </style></head>
<body><p><a href="http://item.jd.com/10860514.html" class="solid">Professional Adobe Flex 3</a>
中文版书名：Adobe Flex 3 高级编程</p>
  <p><a href="http://item.jd.com/10079681.html" class="dashed">Professional WCF Programming</a>
中文版书名：WCF 高级编程</p>
  <p><a href="http://item.jd.com/12033884.html" class="dashed">Professional Swift</a> 中文版书
名：Swift 2.2 高级编程</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-image 属性负责为元素

设置背景图像，元素的背景占据了元素的全部尺寸，包括内边距和边框，但不包括外边距，默认背景图像位于元素的左上角，并在水平和垂直方向上重复；linear-gradient 用于创建线性渐变效果，由于变化距离较小，效果不特别明显。linear-gradient 的语法格式如下。

```
linear-gradient(direction, color-stop1, color-stop2, …);
```

其中，direction 默认为 to bottom，即从上向下的渐变；stop 表示颜色的分布位置，默认均匀分布，如果有 3 个颜色，则各个颜色的 stop 均为 33.33%。

此实例的源文件名是 myHtmlB124.html。

483　以红色波浪线代替显示超链接的下画线

此实例主要通过设置超链接的 text-decoration-style 属性实现以波浪线代替显示超链接的下画线。当在火狐浏览器(Mozilla Firefox)中显示该页面时将以红色的波浪线显示 3 个超链接，如图 483-1 所示。有关此实例的主要代码如下。

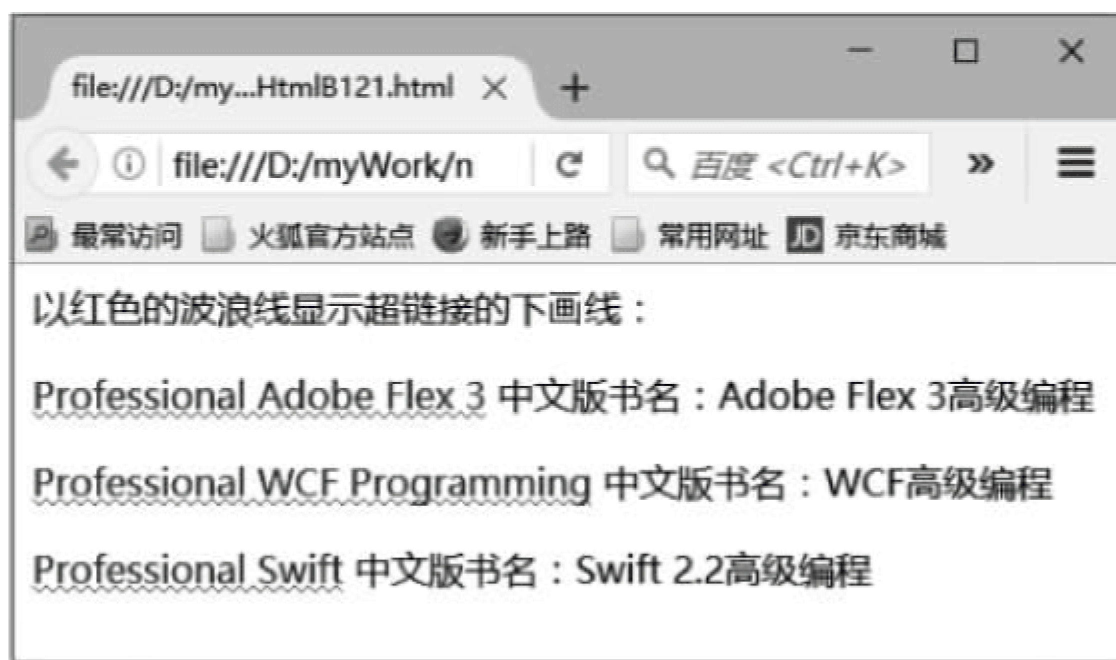


图　483-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a { text-decoration-style: wavy;text-decoration-color: red;
      text-decoration-line: underline; }
 </style></head>
<body>以红色的波浪线显示超链接的下画线：
  <p><a href="http://item.jd.com/10860514.html">Professional Adobe Flex 3</a>中文版书名：Adobe
Flex 3 高级编程</p>
  <p><a href="http://item.jd.com/10079681.html">Professional WCF Programming</a>中文版书名：WCF
高级编程</p>
  <p><a href="http://item.jd.com/12033884.html">Professional Swift</a>中文版书名：Swift 2.2 高级
编程</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-decoration-style：wavy 用于设置装饰线的风格为波浪线。除了火狐浏览器(Mozilla Firefox)，其他浏览器目前都不支持 text-decoration-style 属性。在 CSS3 中，text-decoration-style 属性规定线条如何显示，该属性的语法格式如下。

```
text-decoration-style: solid|double|dotted|dashed|wavy|initial|inherit
```

其中，solid 为默认值，表示线条将显示为单线；double 表示线条将显示为双线；dotted 表示线条

将显示为点状线；dashed 表示线条将显示为虚线；wavy 表示线条将显示为波浪线。

此实例的源文件名是 myHtmlB121.html。

484 为指定的超链接同时设置下画线和删除线

此实例主要为超链接的 text-decoration 属性设置 underline 和 line-through 两个属性值，从而实现在超链接上同时显示下画线和删除线。当在 Google Chrome 浏览器中显示该页面时，后面 3 个超链接将同时显示下画线和删除线，如图 484-1 所示。有关此实例的主要代码如下。

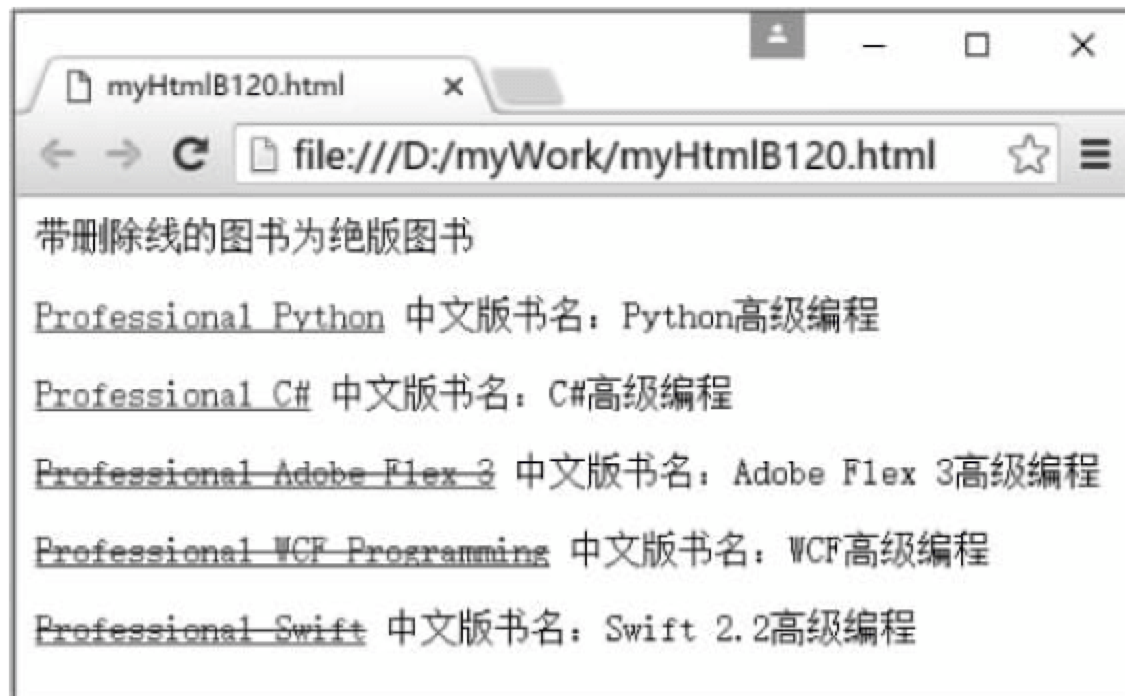


图 484-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a[href*=jd] { text-decoration: underline line-through; }
 </style></head>
<body>带删除线的图书为绝版图书
  <p><a href="https://www.amazon.cn/dp/B01N8TTN52/ref=sr_1_18?s=books&ie=UTF8&qid=1487037038&sr=1-18&keywords=Professional">Professional Python</a>中文版书名：Python高级编程</p>
  <p><a href="https://www.amazon.cn/dp/B00S4HCQT4/ref=sr_1_20?s=books&ie=UTF8&qid=1487037038&sr=1-20&keywords=Professional">Professional C#</a>中文版书名：C#高级编程</p>
  <p><a href="http://item.jd.com/10860514.html">Professional Adobe Flex 3</a>中文版书名：Adobe Flex 3 高级编程</p>
  <p><a href="http://item.jd.com/10079681.html">Professional WCF Programming</a>中文版书名：WCF 高级编程</p>
  <p><a href="http://item.jd.com/12033884.html">Professional Swift</a>中文版书名：Swift 2.2 高级编程</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-decoration：underline line-through 用于为指定的超链接同时设置删除线和下画线。在 CSS3 中，text-decoration 属性规定添加到文本的修饰，修饰的颜色由 color 属性设置。这个属性允许对文本设置某种效果，例如下画线。如果后代元素没有自己的装饰，在祖先元素上设置的装饰会延伸到后代元素中。text-decoration 属性值通常为一个，但也可以像实例这样同时添加两个及两个以上的属性值。text-decoration 属性的语法格式如下。

```
text-decoration: none | underline | blink | overline | line-through
```

其中，none 为默认值，表示定义标准的文本；underline 表示定义文本下的一条线；overline 表示定义文本上的一条线；line-through 表示定义穿过文本的一条线；blink 表示定义闪烁的文本。

此实例的源文件名是 myHtmlB120.html。

485　通过 start 自定义有序列表的开始编号

此实例主要通过设置有序列表的 start 属性实现自定义有序列表的开始编号。当在浏览器中显示该页面时将按照默认值对有序列表的列表项进行编号，如图 485-1 所示。单击“自定义序号的有序列表”按钮，则有序列表将以 13 作为开始编号，并对其他的列表项进行连续编号，如图 485-2 所示。有关此实例的主要代码如下。



图　485-1



图　485-2

```
<!doctype html><html><head><meta charset = UTF - 8 >
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $ (document).ready(function() {
   $ ("#myBtnstart").click(function() {//自定义序号的有序列表
    $ ("#myOl").prop("start",13)  });});
 </script></head>
<body><p><input type = "button" value = "自定义序号的有序列表" id = "myBtnstart" style = "width:
190px"></p>
  <ol id = "myOl" >
    <li>清华大学</li><li>北京大学</li><li>上海交通大学</li><li>浙江大学</li>
    <li>南京大学</li><li>中国人民大学</li><li>中山大学</li></ol></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，有序列表的 start 属性是 HTML5 新增的属性，该属性规定有序列表的开始编号。如果不设置此属性，有序列表则以 1 作为开始编号。

此实例的源文件名是 myHtmlA028.html。

486　通过 reversed 实现有序列表的倒序编号

此实例主要通过设置有序列表的 reversed 属性实现对有序列表进行倒序编号。当在浏览器中显示该页面时将按照升序对有序列表的列表项进行编号，如图 486-1 所示。单击“对有序列表进行倒序编号”按钮，则有序列表的列表项将按照倒序进行编号，如图 486-2 所示。有关此实例的主要代码如下。



图 486-1



图 486-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnreversed").click(function() {//对有序列表进行倒序编号
      $("#myOl").prop("reversed",true); });});
 </script></head>
<body><p><input type="button" value="对有序列表进行倒序编号" id="myBtnreversed" style="width:
190px"></p>
  <ol id="myOl" >
    <li>清华大学</li><li>北京大学</li><li>上海交通大学</li><li>浙江大学</li>
    <li>南京大学</li><li>中国人民大学</li><li>中山大学</li></ol></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，有序列表的 reversed 属性是 HTML5 新增的属性，该属性对列表进行降序排列。如果不设置此属性，有序列表将按照升序进行排列。

此实例的源文件名是 myHtmlA029.html。

487 通过计数器对目录的章节进行自动编号

此实例主要通过设置自动编号的相关属性 counter-reset、counter-increment、content 实现对图书目录的章节进行自动编号。当在 Google Chrome 浏览器中显示该页面时对章节进行自动编号的图书目录如图 487-1 所示。有关此实例的主要代码如下。



图 487-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
    html { color: #444;font: 16px/1.6 微软雅黑, "Microsoft YaHei";}
    ul { list-style: none; padding-left: 3em; }
    .menu { counter-reset: chapter; }
    .menu>li { counter-increment: chapter;counter-reset: section;
               font-size: 20px;   }
    .menu>li:before { content: "第" counter(chapter) "章";
                      padding-right: 0.5em; }
    .menu ul>li { counter-increment: section; font-size: 16px;}
    .menu ul>li:before { content: "第" counter(section) "节";
                         padding-right: 0.5em; }
 </style></head>
<body><ul class="menu">
 <li>上下文管理器<ul><li>上下文管理器的定义</li>
         <li>上下文管理器的语法</li><li>资源清理</li></ul></li>
 <li>生成器<ul><li>理解生成器</li>
         <li>迭代对象与迭代器</li><li>StopIteration异常</li>
         <li>何时编写生成器</li><li>分块计算数据</li></ul></li>
 <li>正则表达式<ul><li>使用正则表达式的原因</li>
         <li>match对象</li><li>命名分组</li>
         <li>点匹配换行符</li></ul></li></ul>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，counter-increment：chapter表示对chapter按照1的增量进行自增编号，如果将此行修改为counter-increment：chapter 4，则将显示第4章、第8章、第12章。在CSS中，counter-increment属性用于设置某个选择器每次出现的计数器增量，默认增量是1。使用这个属性，计数器可以递增(或递减)某个值，可以是正值或负值，如果没有提供number值，则默认为1。counter-reset：chapter表示chapter的起始点为第1章，如果将此行修改为counter-reset：chapter 3，则第1章将显示为第4章。在CSS中，counter-reset属性用于设置某个选择器出现次数的计数器的值，默认为0。使用这个属性，计数器可以设置或重置为任何值，可以是正值或负值，如果没有提供number，则默认为0。content属性主要用于和before选择器及after选择器配合使用来插入生成内容。

此实例的源文件名是myHtmlB210.html。

488 通过计数器自动统计所选择复选框的数量

此实例主要通过设置元素的counter-reset、counter-increment、content等属性实现使用纯CSS自动统计用户选择的复选框数量。当在Google Chrome浏览器中显示该页面时，如果选择其中的6家公司，则统计结果如图488-1所示；如果取消选择其中的两家公司，则统计结果如图488-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  body {counter-reset: myCounter;}
  input:checked { counter-increment: myCounter;}
  .total::after { color:red; content: counter(myCounter); }
 </style></head>
<body><strong>下面的中国500强公司哪几家与你有关?</strong>
```

```
<ol><li><input type="checkbox" id="myCounter1"><label for="myCounter1">中国邮政集团公司</label>
</li>
 <li><input type="checkbox" id="myCounter2"><label for="myCounter2">中国平安保险(集团)股份有限
公司</label></li>
 <li><input type="checkbox" id="myCounter3"><label for="myCounter3">中国农业银行股份有限公司
</label></li>
 <li><input type="checkbox" id="myCounter4"><label for="myCounter4">中国石油天然气集团公司
</label></li>
 <li><input type="checkbox" id="myCounter5"><label for="myCounter5">上海汽车集团股份有限公司
</label></li>
 <li><input type="checkbox" id="myCounter6"><label for="myCounter6">中国兵器装备集团公司
</label></li>
 <li><input type="checkbox" id="myCounter7"><label for="myCounter7">中国电信集团公司
</label></li>
 <li><input type="checkbox" id="myCounter8"><label for="myCounter8">中国中信集团有限公司
</label></li>
 <li><input type="checkbox" id="myCounter9"><label for="myCounter9">中国人民保险集团股份有限公
司</label></li>
 <li><input type="checkbox" id="myCounter10"><label for="myCounter10">中国远洋海运集团有限公司
</label></li>
</ol>
你总共选择了<strong class="total"></strong>家公司</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,counter-reset 属性用于设置某个选择器出现次数的计数器的值,默认为 0;counter-increment 属性用于设置某个选择器每次出现的计数器增量,默认增量是 1,使用这个属性,计数器可以递增(或递减)某个值,可以是正值或负值,如果没有提供数字值,则默认为 1;after 选择器用于在元素之后添加内容,这个选择器允许在元素内容的最后插入生成内容。

此实例的源文件名是 myHtmlB136.html。



图 488-1



图 488-2

489 在同级子元素前面插入自增的数字编号

此实例主要通过在 content 属性中采用 counter(myCounter)实现在同级子元素的前面插入自增的数字编号。当在 Google Chrome 浏览器中显示该页面时,每本书名的前面将自动添加一个自增的

连续数字编号，如图 489-1 所示。有关此实例的主要代码如下。



图　489-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  p:before{content: "第"counter(myCounter)"名.";
           font-size: 16px; font-weight: bold;}
  p {padding-left: 8px; width: 380px; margin: 10px; background-color: lightgreen;
     border-radius: 5px;counter-increment: myCounter;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
  <p>科学的旅程</p>
  <p>万物解释者：复杂事物的极简说明书</p>
  <p>地球与太空：美国宇航局 NASA 珍贵摄影集</p>
  <p>迷人的数学：315 个烧脑游戏玩通数学史</p>
  <p>无言的宇宙：隐藏在 24 个数学公式背后的故事</p>
  <p>中国国家地理百科全书</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content："第"counter(myCounter)"名."用于在同级子元素的前面插入自增的连续数字编号。在 content 属性中使用 counter 属性值来针对多个同级子元素追加连续的编号，使用方法如下。

```
<元素>:before{
  content:counter(计数器名);
}
```

计数器名可以任意命名，此外还需要在元素的样式中追加对元素的 counter-increment 属性的指定，为了实现连续编号，应该将 counter-increment 属性的属性值设定为 before 选择器或 after 选择器的 counter 属性值中所指定的计数器名，格式如下。

```
<元素>{counter-increment:before 选择器或 after 选择器中指定的计数器名}
```

此实例的源文件名是 myHtmlB028.html。

490　在同级子元素前面插入自增的字母编号

此实例主要通过在 content 属性中采用 counter(myCounter,lower-alpha)实现在同级子元素的前面插入自增的字母编号。当在 Google Chrome 浏览器中显示该页面时，每本书名的前面将自动添加一个自增的连续小写字母编号，如图 490-1 所示。有关此实例的主要代码如下。



图 490-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*使用连续的小写字母对同级子元素进行编号*/
  p:before{ content: counter(myCounter,lower-alpha)".";
            font-size: 16px;font-weight: bold;}
  p { padding-left: 8px; width: 380px; margin: 10px;background-color: lightgreen;
     border-radius: 5px; counter-increment: myCounter;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
  <p>科学的旅程</p>
  <p>万物解释者：复杂事物的极简说明书</p>
  <p>地球与太空：美国宇航局 NASA 珍贵摄影集</p>
  <p>迷人的数学：315 个烧脑游戏玩通数学史</p>
  <p>无言的宇宙：隐藏在 24 个数学公式背后的故事</p>
  <p>中国国家地理百科全书</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content: counter (myCounter, lower-alpha)"."用于实现在同级子元素的前面插入自增的小写字母编号。其中，lower-alpha 表示编号种类是小写字母，编号种类可以使用 list-style-type 属性值来指定，list-style-type 属性值的种类取值有多种，其中 disc 为实心圆(CSS1)、circle 为空心圆(CSS1)、square 为实心方块(CSS1)、decimal 为阿拉伯数字(CSS1)、lower-roman 为小写罗马数字(CSS1)、upper-roman 为大写罗马数字(CSS1)、lower-alpha 为小写英文字母(CSS1)、upper-alpha 为大写英文字母(CSS1)、none 为不使用项目符号(CSS1)、armenian 为传统的亚美尼亚数字(CSS2)、cjk-ideographic 为浅白的表意数字(CSS2)、georgian 为传统的乔治数字(CSS2)、lower-greek 为基本的希腊小写字母(CSS2)、hebrew 为传统的希伯莱数字(CSS2)、hiragana 为日文平假名字符(CSS2)、hiragana-iroha 为日文平假名序号(CSS2)、katakana 为日文片假名字符(CSS2)、katakana-iroha 为日文片假名序号(CSS2)、lower-latin 为小写拉丁字母(CSS2)、upper-latin 为大写拉丁字母(CSS2)。

此实例的源文件名是 myHtmlB029.html。

491 在同级子元素前面插入自增的罗马数字

此实例主要通过在 content 属性中采用 counter(myCounter, upper-roman)"."实现在同级子元素的前面插入自增的大写罗马数字。当在 Google Chrome 浏览器中显示该页面时，每本书名的前面将

自动添加一个自增的连续编号的大写罗马数字,如图 491-1 所示。有关此实例的主要代码如下。



图 491-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*使用连续的大写罗马数字对同级子元素进行编号*/
  p:before{ content: counter(myCounter,upper-roman)".";
           font-size: 16px; font-weight: bold; }
  p { padding-left: 8px; width: 380px; margin: 10px; background-color: lightseagreen;border-radius:
5px; counter-increment: myCounter; }
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
  <p>科学的旅程</p>
  <p>万物解释者: 复杂事物的极简说明书</p>
  <p>地球与太空: 美国宇航局 NASA 珍贵摄影集</p>
  <p>迷人的数学: 315 个烧脑游戏玩通数学史</p>
  <p>无言的宇宙: 隐藏在 24 个数学公式背后的故事</p>
  <p>中国国家地理百科全书</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,content: counter (myCounter, upper-roman)"."用于实现在同级子元素的前面插入自增的连续编号的大写罗马数字。其中,upper-roman 表示编号种类是大写罗马数字,编号种类可以使用 list-style-type 属性值来指定。

此实例的源文件名是 myHtmlB030.html。

492 在父子关系的结构中对子元素嵌套编号

此实例主要定义两个计数器,并在 counter-reset 属性中重置 mySubCounter 子级计数器,从而实现对父、子两级中的元素分别进行编号。当在 Google Chrome 浏览器中显示该页面时,月份采用大写罗马数字进行计数,每月份的上榜书名采用数字进行计数,如图 492-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*使用连续的大写罗马数字对父级子元素进行编号*/
  h2:before{ content: counter(myCounter,upper-roman)".";
            font-size: 16px; font-weight: bold;}
```

```
    h2{ counter-increment:myCounter; counter-reset:mySubCounter; }
    /*使用连续的数字对子级子元素进行编号*/
    p:before{ content:counter(mySubCounter)".";  }
    p{ padding-left: 8px;width: 380px; margin: 10px; background-color: lightseagreen;
      border-radius: 5px;counter-increment: mySubCounter; }
  </style></head>
 <body><h2>10月份新上榜科技图书</h2>
   <p>博物学家的神秘动物图鉴</p><p>数学家讲解小学数学</p>
   <h2>11月份新上榜科技图书</h2>
   <p>图解时间简史</p><p>世界上最老最老的生命</p><p>化石知道生命的旅程</p>
   <h2>12月份新上榜科技图书</h2>
   <p>科学的旅程</p><p>万物解释者: 复杂事物的极简说明书</p>
   <p>地球与太空: 美国宇航局NASA珍贵摄影集</p>
   <p>迷人的数学: 315个烧脑游戏玩通数学史</p>
   <p>无言的宇宙: 隐藏在24个数学公式背后的故事</p>
   <p>中国国家地理百科全书</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,counter-reset属性用于设置某个选择器出现次数的计数器的值,默认为0。使用这个属性,计数器可以设置或重置为任何值,可以是正值或负值(例如counter-reset:mySubCounter -10)。如果没有提供number,则默认为0。

此实例的源文件名是myHtmlB031.html。



图 492-1

493 仿照图书目录对多级结构元素嵌套编号

此实例主要定义3个计数器,并在counter-reset属性中重置(例如mySubCounter)子级计数器,从而实现仿照图书目录对多级结构的元素进行嵌套编号。当在Google Chrome浏览器中显示该页面时,对三级目录进行自动嵌套编号的效果如图493-1所示。有关此实例的主要代码如下。



图　493-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  h2:after,h3:after,h4:after{ content:"简介"; }
  h2:before{ content: counter(myCounter) '. ';}
  h2{counter-increment: myCounter;counter-reset: mySubCounter;}
  h3:before{ content:counter(myCounter) '-' counter(mySubCounter) '. ';}
  h3{counter-increment: mySubCounter;
     counter-reset: mySubSubCounter; margin-left: 40px;}
  h4:before{content:counter(myCounter) '-' counter(mySubCounter) '-'
            counter(mySubSubCounter)'. ';}
  h4{counter-increment: mySubSubCounter; margin-left: 40px;}
 </style></head>
<body>
  <h2>北京市</h2><h3>海淀区</h3>
   <h4>八里庄街道办事处</h4> <h4>万寿路街道办事处</h4><h4>清河街道办事处</h4>
  <h3>昌平区</h3><h4>城北街道办事处</h4><h4>回龙观街道办事处</h4>
  <h2>重庆市</h2><h3>渝北区</h3><h4>龙溪街道办事处</h4><h4>回兴街道办事处</h4> <h4>宝圣湖
街道办事处</h4><h4>人和街道办事处</h4>
  <h3>北碚区</h3><h4>北温泉街道办事处</h4><h4>朝阳街道办事处</h4>
  <h3>巴南区</h3><h3>江北区</h3></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，counter-reset 属性用于设置某个选择器出现次数的计数器的值，默认为 0。当需要对多级元素进行嵌套编号时通常需要设置此属性，否则会出现编号混乱。h2:after,h3:after,h4:after{ content:"简介"; }用于在每行文字的后面添加“简介”；content: counter(myCounter) '. '用于设置一级目录的格式，例如“1. 北京市简介”；content: counter(myCounter) '-' counter(mySubCounter) '. '用于设置二级目录的格式，例如“1-1. 海淀区简

介”；content：counter(myCounter) '-' counter(mySubCounter) '-'counter(mySubSubCounter)'. '用于设置三级目录的格式，例如“1-1-1. 八里庄街道办事处简介”。

此实例的源文件名是 myHtmlB032.html。

494 在字符串两边添加嵌套的对称文字符号

此实例主要设置 quotes 属性值，并使用 content 属性的 open-quote 属性值和 close-quote 属性值，从而实现在字符串的两边添加嵌套的对称文字符号。当在 Google Chrome 浏览器中显示该页面时，每本书名的前后将自动添加书名号，如图 494-1 所示。有关此实例的主要代码如下。



图 494-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  p:before{content: open-quote;} /* 前面的书名号 */
  p:after{content: close-quote;} /* 后面的书名号 */
  p { padding-left: 8px; width: 380px; margin: 10px; background-color: lightgreen;
     border-radius: 5px; quotes:"【" "】";}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
  <p>科学的旅程</p>
  <p>万物解释者：复杂事物的极简说明书</p>
  <p>地球与太空：美国宇航局 NASA 珍贵摄影集</p>
  <p>迷人的数学：315 个烧脑游戏玩通数学史</p>
  <p>无言的宇宙：隐藏在 24 个数学公式背后的故事</p>
  <p>中国国家地理百科全书</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，content 属性的 open-quote 属性值和 close-quote 属性值用于在字符串两边添加诸如括号、单引号、双引号之类的嵌套符号。其中，open-quote 属性值用于添加开始的嵌套符号，close-quote 属性值用于添加结尾的嵌套符号。在使用这两个属性值之前应该首先在元素（例如 p 元素）样式的 quotes 属性中指定使用什么嵌套符号。此外，当需要添加双引号时需要使用转义字符“\”。

此实例的源文件名是 myHtmlB033.html。

495 使用图像替换无序列表的列表项标记

此实例主要通过设置 list-style-image 属性实现使用自定义图像来替换无序列表的列表项标记。当在 Google Chrome 浏览器中显示该页面时，每个列表项前面均有一个自定义图像，如图 495-1 所示。有关此实例的主要代码如下。



图 495-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  ul { list-style-image: url('img/B184A.png'); }
  li { list-style-type: none;padding-left: 8px;width:340px; margin:10px; text-align: left; }
 </style></head>
<body><center><ul><li>科学的旅程</li>
    <li>万物解释者：复杂事物的极简说明书</li>
    <li>地球与太空：美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学：315 个烧脑游戏玩通数学史</li>
  <li>无言的宇宙：隐藏在 24 个数学公式背后的故事</li></ul></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，list-style-image：url('img/B184A.png')用于实现使用自定义图像设置列表项的标记，该代码也可以放置在 li 元素对应的 CSS 类中实现相同的外观效果，但是推荐放置在 ul 元素对应的 CSS 类中。在 CSS 中，list-style-image 属性使用图像来替换列表项的标记，此属性值指定有序或无序列表项标志的图像，图像相对于列表项内容的放置位置通常使用 list-style-position 属性控制，为防止意外发生，通常应规定一个 list-style-type 属性以防图像不可用。list-style-image 属性的语法格式如下。

```
list-style-image:URL|none|inherit;
```

其中，URL 表示图像的路径；none 为默认值，表示无图形被显示；inherit 规定应该从父元素继承 list-style-image 属性的值。

此实例的源文件名是 myHtmlB184.html。

496 在多个条目中获取使用鼠标选择的条目

此实例主要通过使用 jquery-ui.min.js 和 jquery-ui.min.css 的 selectable()方法实现在组条目中获取使用鼠标选择的多个条目。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标选择第

1、2、3、5 几个条目，则将在下面显示选择结果，如图 496-1 所示；如果使用鼠标取消选择第 3 个条目，则在下面显示的选择结果中将不显示第 3 个条目的名称，如图 496-2 所示。当然，所有条目可以任意测试。有关此实例的主要代码如下。



图 496-1



图 496-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script>
 <script src="js/jquery-ui.min.js"></script><script type="text/javascript">
  $(function() {
   $( "#myItems" ).selectable({                        //获取选择的多个条目
    stop: function() {
     var myResult = $( "#myResult" ).empty();
     $( ".ui-selected", this ).each(function() {
      var myIndex = $( "#myItems li" ).index( this );
      //显示选择结果
      myResult.append("<br>" + $ ( "#myItems li" )[myIndex].innerText );
  });} });});
  </script>
  <link rel="stylesheet" href="css/jquery-ui.min.css"><style type="text/css">
    *{ margin:0; padding: 0; }
    .myContainer { margin:20px 30px;}                  /*设置盒子的基本样式*/
    ul,ul li{list-style: none;}                        /*取消无序列表的默认样式*/
    ul li{ padding:5px; margin-top:5px; margin-bottom:5px;}
    ul li.ui-selected { background:cyan; }             /*设置选择条目的样式*/
    ul li:hover{ cursor:default;}
 </style></head>
<body><div class="myContainer" align="center"><ul id="myItems">
  <li class="ui-widget-content">Web 开发入门经典</li>
  <li class="ui-widget-content">精通 HTML5 + CSS3 + JavaScript 网页设计</li>
  <li class="ui-widget-content">HTML5 + CSS3 从入门到精通</li>
  <li class="ui-widget-content">HTML5 从入门到精通</li>
  <li class="ui-widget-content">HTML5 游戏开发</li></ul>
 <p><span>您已经选择了：</span><span id="myResult"></span></p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，selectable()方法用于处理使用鼠标

选择的单个元素或一组元素，参数中的 stop 表示在停止鼠标动作后执行这部分代码。需要说明的是，selectable()方法是可选择小部件(Selectable Widget)的方法，因此在使用 selectable()方法时必须添加 jquery-ui. min. js 和 jquery-ui. min. css 两个文件。

此实例的源文件名是 myHtmlA160. html。

497　使用伪元素重置默认的颜色选择器按钮

此实例主要定制伪元素::-webkit-color-swatch-wrapper 和::-webkit-color- swatch，从而实现重置默认的颜色选择器按钮。当在 Google Chrome 浏览器中显示该页面时，下面一个按钮是默认的颜色选择器按钮，上面一个是采用渐变色定制的颜色选择器按钮，如图 497-1 所示，单击任意一个颜色选择器按钮都将弹出“颜色”对话框。有关此实例的主要代码如下。



图　497-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myColorDlg").change(function() {     //选择颜色
    $("body").css("background-color", this.value);
   });
   $("#ColorDlg").change(function() {       //选择颜色
    $("body").css("background-color", this.value); });});
 </script>
 <style type="text/css">
  .myColor{ }
  /*重新定制颜色选择器按钮的外层*/
  .myColor::-webkit-color-swatch-wrapper { width:100px; height:50px; border-radius: 5px; box-shadow:0 15px 10px rgba(13,30,4,0.8); position:relative; left: -10px; top:-5px; border: 0px solid #777; background-image: -webkit-gradient(linear, 0 0, 100% 0,from(red), color-stop(15%, orange), color-stop(30%, yellow), color-stop(50%, green), color-stop(65%, darkcyan), color-stop(80%, blue), to(purple)); }
  /*隐藏内层颜色显示部分*/
  .myColor::-webkit-color-swatch { height:21px; position:relative; left: 140px; top:15px;display:none; }
 </style></head>
<body><center>
   单击自定义颜色选择器按钮更改背景颜色: <input id="myColorDlg" type="color"  class="myColor"/>
<br><br><br>
   单击默认颜色选择器按钮更改背景颜色: <input id="ColorDlg" type="color" /> </center></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-color-swatch- wrapper 伪元素对应默认的颜色选择器按钮的外层，::-webkit-color-swatch 伪元素对应默认的颜色选择器按钮的内层，因此定制此颜色选择器按钮就是设置这两部分的样式。这两部分可以同时显示，此实例只是设置内层部分的 display: none 属性隐藏了内层。按照同样的思路也可以只显示内层，不显示外层。

此实例的源文件名是 myHtmlB165.html。

498　使用伪元素定制日期选择器的部分样式

此实例主要定制伪元素::-webkit-datetime-edit-fields-wrapper、::-webkit- datetime-edit-text、::-webkit- datetime-edit-year-field、::-webkit-datetime-edit-month-field、::-webkit-datetime-edit-day-field，从而实现定制日期选择器的部分样式。当在 Google Chrome 浏览器中显示该页面时，“出生日期:”是默认的日期选择器，“入学日期:”是采用上述伪元素定制的日期选择器，如图 498-1 所示。有关此实例的主要代码如下。

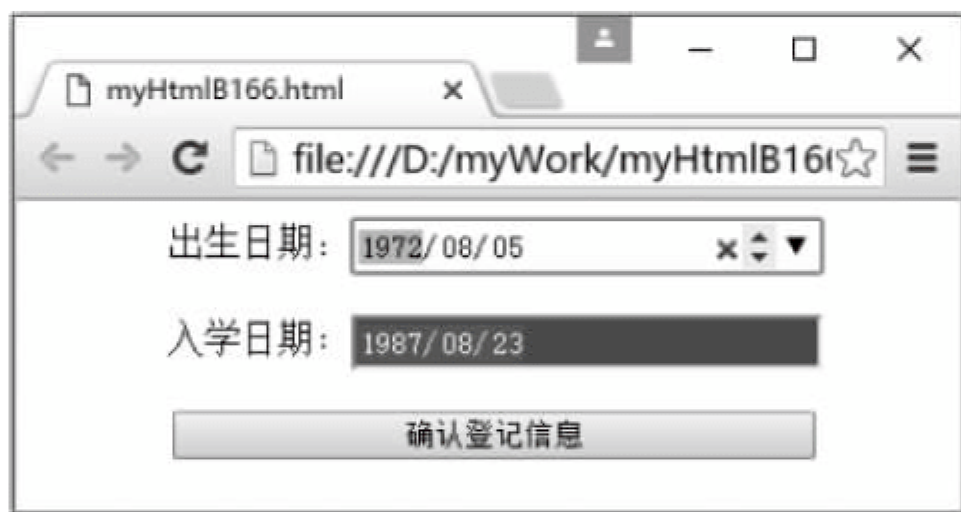


图　498-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtn").click(function() {//确认登记信息
    alert("您登记的入学日期是: "+$("#myDate").prop("value"));  }) });
 </script>
 <style type="text/css">
  .myDate {background: green; }
  /*重新定制日期选择器的可编辑部分背景颜色*/
  .myDate::-webkit-datetime-edit-fields-wrapper { background: red;}
  /*重新定制日期选择器的可编辑部分文本颜色,即两条左斜线*/
  .myDate::-webkit-datetime-edit-text {color: yellow; }
  /*重新定制日期选择器的可编辑部分的年、月、日文本颜色*/
  .myDate::-webkit-datetime-edit-year-field,
  .myDate::-webkit-datetime-edit-month-field,
  .myDate::-webkit-datetime-edit-day-field { color: white;}
  input{width: 200px;}
  button{width: 280px;}
 </style></head>
<body><center><form style="width: 400px;">
 出生日期: <input type="date" id="myBirth" value="1972-08-05"/><br><br>
 入学日期: <input type="date" id="myDate" value="1987-08-23" class="myDate"/><br><br><button
type="button" id="myBtn">确认登记信息</button></form></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，. myDate::-webkit- datetime-edit-text{ color: yellow;}表示通过伪元素::-webkit-datetime-edit-text 设置日期选择器的可编辑文本部分的颜色为黄色。下面 8 个伪元素是 webkit 专门用于定制输入(或选择)日期的，即::-webkit-datetime-edit、::-webkit-datetime-edit-fields-wrapper、::-webkit-datetime-edit-text、::-webkit-datetime-edit-month-field、::-webkit-datetime-edit-day-field、::-webkit-datetime-edit-year-field、::-webkit-inner-spin-button、::-webkit-calendar-picker-indicator。

此实例的源文件名是 myHtmlB166. html。

499　使用伪元素重置文件上传按钮的样式

此实例主要通过定制伪元素::-webkit-file-upload-button 重置文件上传按钮的样式。当在 Google Chrome 浏览器中显示该页面时，第一个“选择文件”按钮是使用伪元素::-webkit-file-upload-button 重新定制后的文件上传按钮的样式，第二个“选择文件”按钮是默认的文件上传按钮的样式，单击第一个“选择文件”按钮即可选择图像文件，并显示在页面中，如图 499-1 所示。有关此实例的主要代码如下。



图　499-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function showImage() {                        //当从本地计算机中选择文件以后显示图像
    var myFile=$("input").get(0).files[0];
   if (!/image\/\w+/.test(myFile.type)) {   //检查是否为图像文件
    alert("请确保文件为图像类型"); return false;   }
   var myReader = new FileReader();
   myReader.readAsBinaryString(myFile);       //以二进制形式读取图像文件
   myReader.onload=function(f) {
    var myDiv = document.getElementById("myDiv");
    var mySrc="data:"+myFile.type+";base64,"+window.btoa(this.result);
    myDiv.innerHTML='<img id="myImage" src="'+mySrc+'"/>'   } }
 </script>
```

```
<style type = "text/css">
   .myFile{   }
   /* 重新定制文件上传按钮的样式 */
   .myFile::-webkit-file-upload-button { width: 100%;}
   input{width:395px;}
   img{border-radius: 5px;width:400px;height:250px;}
 </style></head>
<body><p><input type = "file" onchange = "showImage()" class = "myFile"/></p>
  <p><input type = "file" /></p><div id = "myDiv"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，.myFile::-webkit-file-upload-button { width：100%；}表示在::-webkit-file-upload-button 伪元素中设置按钮的宽度是全部充满状态，即隐藏“未选择任何文件”等提示文本。当然，也可以定制 background-color 等其他 CSS 样式。

此实例的源文件名是 myHtmlB167.html。

500 使用伪元素去掉数字选择器的上、下按钮

此实例主要通过在伪元素::-webkit-textfield-decoration-container、::-webkit-inner-spin-button、::-webkit-outer-spin-button 中设置 CSS 样式实现去掉数字选择器的上、下按钮。当在 Google Chrome 浏览器中显示该页面时，“公司女工人数：”是默认的数字选择器，因此可以通过单击右端的上、下按钮选择数字；“公司男工人数：”是定制的数字选择器，右端没有选择数字的上、下按钮，并且背景呈现灰色，如图 500-1 所示。有关此实例的主要代码如下。



图 500-1

```
<!doctype html><html><head><meta charset = UTF-8>
 <style type = "text/css">
  .myNumber{   }
  /* 设置背景颜色为浅灰色 */
  .myNumber::-webkit-textfield-decoration-container{
            background-color: lightgray;}
  /* 去掉右端的上、下按钮 */
  .myNumber::-webkit-inner-spin-button,.myNumber::-webkit-outer-spin-button {
            -webkit-appearance: none; }
  button{width:395px;}
  input{width:130px;}
 </style></head>
<body><center><form>公司男工人数: <input type = "number" name = "myAge" min = "5" max = "1000" class =
"myNumber"/>(5-1000 的数字)<br><br>
    公司女工人数: <input type = "number" name = "myAge" min = "1" max = "1000"/>(1-1000 的数字)<br>
<br><button type = "submit" value = "提交">提交注册信息</button> </form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-textfield-decoration-container、::-webkit-inner-spin-button、::-webkit-outer-spin-button 是 input[type=number]数字选择器在 Google Chrome 浏览器中被支持的 3 个伪元素，如果需要定制数字选择器的样式，直接在其中添加 CSS 代码即可。

此实例的源文件名是 myHtmlB168.html。

501　使用伪元素重置 range 元素的滑块、滑槽

此实例主要通过在伪元素::-webkit-slider-runnable-track、::-webkit-slider-thumb 中设置 CSS 样式实现改变 range 元素的默认样式。当在 Google Chrome 浏览器中显示该页面时，"拖动滑块改变图像透明度："是新定制的 range 元素，滑块位于滑槽中，可以通过拖动滑块改变图像的透明度；"拖动滑块改变图像的尺寸："是默认的 range 元素，滑块位于轴心线上，可以通过拖动滑块改变图像的大小，如图 501-1 所示。有关此实例的主要代码如下。



图　501-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    function myFunc() {                          //获取滑块的当前值并转换成 Float
     var myValue = parseFloat($("#myOpacity").prop("value"));
    $("img").css("opacity",myValue);              //根据滑块值重置图像的透明度
    var myHeight = parseFloat($("#myHeight").prop("value"));
    $("img").css("height",myHeight * 100 + "%");  //根据滑块值重置图像的大小
   }
  setInterval(myFunc,1);  });
 </script>
 <style type = "text/css">
  .myRange{  }
  .myRange:: - webkit - slider - runnable - track {     /* 定制滑槽 */
          border: 1px solid black; background: lightgray;}
  /* 定制滑块 */
  .myRange:: - webkit - slider - thumb {margin: 1px;}
```

```
  form{ height: 300px; }
  img{ height: 50 % ; border - radius: 5px;}
 </style></head>
<body><center><form>拖动滑块改变图像透明度: <input type = "range" id = "myOpacity" min = "0" max =
"1" value = "0.5" class = "myRange" step = "0.1"/><br>
  拖动滑块改变图像的尺寸: <input type = "range" id = "myHeight" min = "0" max = "1" value = "0.5" step =
"0.05" /><br><br>
 <img src = "img/B157B.jpg" ></form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,::-webkit-slider- runnable-track 伪元素的 CSS 代码用于设置 range 元素的滑槽样式,::-webkit-slider- thumb 伪元素的 CSS 代码用于设置 range 元素的滑块样式。这两个伪元素是 range 元素在 Google Chrome 浏览器中被支持的两个伪元素,如果需要定制 range 元素的样式,直接在其中添加 CSS 代码即可。

此实例的源文件名是 myHtmlB169.html。

502 使用伪元素重置 progress 元素的样式

此实例主要通过在伪元素::-webkit-progress-inner-element、::-webkit-progress-bar、::-webkit-progress-value 中设置 CSS 样式改变 progress 元素的默认样式。当在 Google Chrome 浏览器中显示该页面时,单击"启动默认进度条"按钮,则默认进度条的进度更新效果如图 502-1 所示;单击"启动定制进度条"按钮,则定制进度条的进度更新效果如图 502-2 所示。有关此实例的主要代码如下。

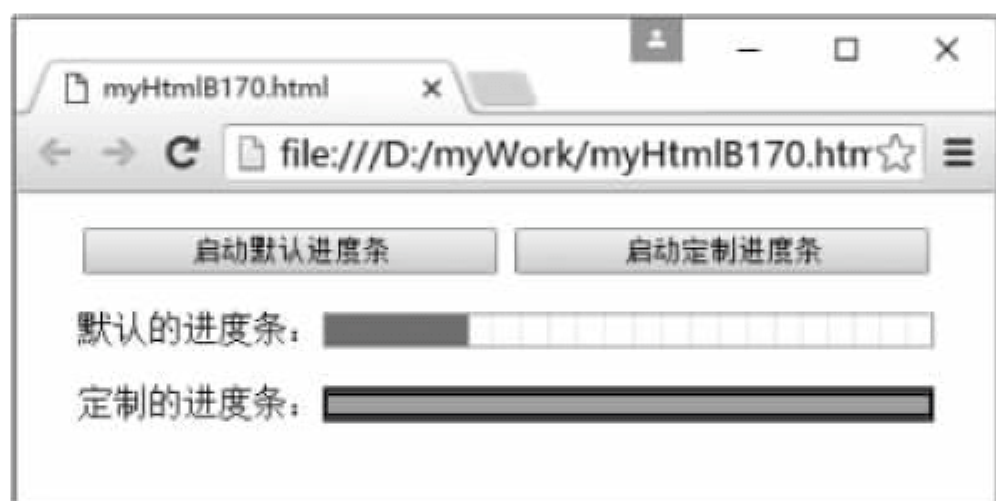


图 502-1

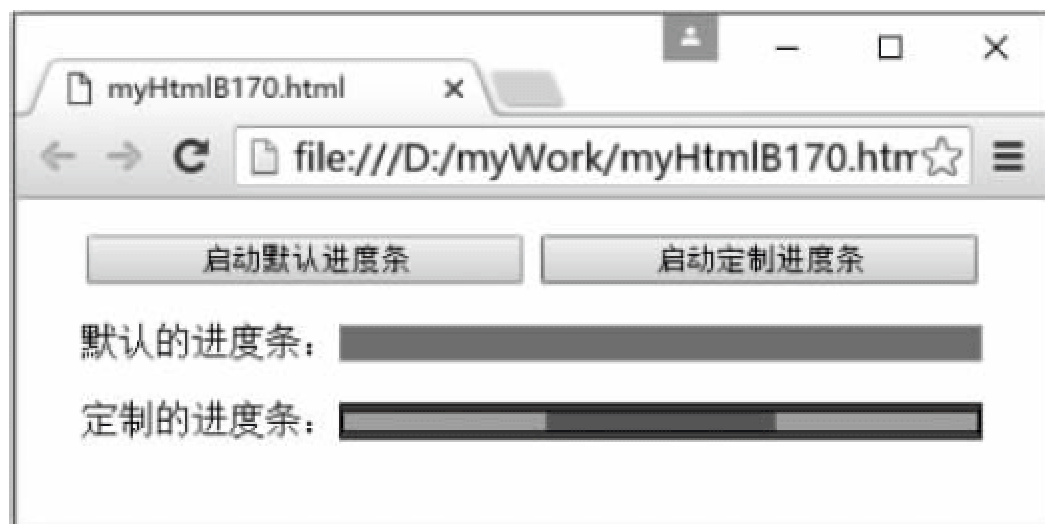


图 502-2

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
  $ (document).ready(function() {
   var myValue = 0; var myTimer;
   function myDefaultBarFunc() { myValue++;
    $ ("#myDefaultBar").val(myValue);
    if (myValue> = $ ("#myDefaultBar").prop("max")) {
     clearInterval(myTimer);
   } }
   $ ("#myBtnDefault").click(function() {        //启动默认进度条
    myValue = 0;
    myTimer = setInterval(myDefaultBarFunc, 100);
   });
   function myCustomBarFunc() { myValue++;
    $ ("#myCustomBar").val(myValue);
    if (myValue> = $ ("#myCustomBar").prop("max")) {
```

```
        clearInterval(myTimer);
      } }
      $ ("#myBtnCustom").click(function() {       //启动定制进度条
       myValue = 0;
       myTimer = setInterval(myCustomBarFunc, 100);  });});
  </script>
  <style type = "text/css">
   progress{ width: 280px;}
   .myProgress {  -webkit-appearance: none; }
   .myProgress::-webkit-progress-inner-element {
               background-color: green; border: 1px solid black; }
   .myProgress::-webkit-progress-bar { border: 1px solid black; background-color: aqua; height:
70%; margin-top: 2px; }
   .myProgress::-webkit-progress-value { background: red;}
   button { width: 190px; }
  </style></head>
 <body><center><form>
   <p><button id = "myBtnDefault" type = "button">启动默认进度条</button>
      <button id = "myBtnCustom" type = "button">启动定制进度条</button></p>
   <p>默认的进度条: <progress value = "0" max = "100" id = "myDefaultBar" ></progress>        </p><p>
 定制的进度条: <progress value = "0" max = "100" id = "myCustomBar" class = "myProgress"></progress></p>
 </form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-progress-inner-element、::-webkit-progress-bar、::-webkit-progress-value 是 progress 元素在 Google Chrome 浏览器中被支持的 3 个伪元素，这 3 个伪元素的层次关系依次是由大到小，由外层到里层，如果需要定制 progress 元素的样式，直接在其中添加 CSS 代码即可。此实例为了区分这 3 层的关系，特别在每一层中分别使用了 green、aqua、red 几种颜色。

此实例的源文件名是 myHtmlB170.html。

503　使用伪元素自定义滚动条的轨道等样式

此实例主要在伪元素::-webkit-scrollbar-track、::-webkit-scrollbar-thumb、::-webkit-scrollbar 中设置 CSS 样式，从而实现改变拥有 overflow 属性的元素的默认滚动条样式。当在 Google Chrome 浏览器中显示该页面时，上面显示的蓝色圆角滚动条是定制的滚动条，下面显示的灰色滚动条是默认的滚动条，如图 503-1 所示。有关此实例的主要代码如下。



图　503-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
   .scrollbar::-webkit-scrollbar { width: 16px;}
   /*定义滚动条的轨道,内阴影*/
   .scrollbar::-webkit-scrollbar-track { -webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
border-radius: 0px; background-color: lightgray; }
   /*定义滑块,内阴影及圆角*/
   .scrollbar::-webkit-scrollbar-thumb {border-radius: 10px; -webkit-box-shadow: inset 0 0 6px
rgba(0, 0, 0, 0.3); background-color: deepskyblue;}
 </style></head>
<body><center><form><p>显示定制的滚动条:<textarea class="scrollbar">台北故宫博物院内收藏有自
南京国立中央博物院筹备处、国立北平故宫博物院和国立北平图书馆等所藏来自北京故宫、沈阳故宫、避暑山
庄、颐和园、静宜园和国子监等处的皇家旧藏。所藏的商周青铜器,历代的玉器、陶瓷、古籍文献、名画碑帖等皆
为稀世之珍。展馆每三个月更换一次展品。截至 2014 年底,馆藏文物达 69.6 万余件文物。</textarea></p>
  <p>显示默认的滚动条:<textarea>台北故宫博物院内收藏有自南京国立中央博物院筹备处、国立北平故宫
博物院和国立北平图书馆等所藏来自北京故宫、沈阳故宫、避暑山庄、颐和园、静宜园和国子监等处的皇家旧藏。
所藏的商周青铜器,历代的玉器、陶瓷、古籍文献、名画碑帖等皆为稀世之珍。展馆每三个月更换一次展品。截
至 2014 年底,馆藏文物达 69.6 万余件文物。</textarea></p></form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,::-webkit-scrollbar-track、::-webkit-scrollbar-thumb、::-webkit-scrollbar 是在 Google Chrome 浏览器中被支持的 3 个伪元素,这些伪元素支持拥有 overflow 属性的区域、列表框、下拉菜单、textarea 等元素的滚动条自定义样式。关于滚动条样式定制的伪元素的功能说明如下。

(1) ::-webkit-scrollbar:用于自定义滚动条整体部分。

(2) ::-webkit-scrollbar-thumb:用于自定义滚动条里面的小方块,能上下左右移动(取决于是垂直滚动条还是水平滚动条)。

(3) ::-webkit-scrollbar-track:用于自定义滚动条的轨道(里面装有 thumb)。

(4) ::-webkit-scrollbar-button:用于自定义滚动条轨道两端的按钮,允许通过单击微调小方块的位置。

(5) ::-webkit-scrollbar-track-piece:用于自定义内层轨道,除去滚动条的中间部分。

(6) ::-webkit-scrollbar-corner:用于自定义边角以及两个滚动条的交汇处。

(7) ::-webkit-resizer:自定义两个滚动条的交汇处用于通过拖动调整元素大小的小控件。

此实例的源文件名是 myHtmlB171.html。

504 使用伪元素定制默认滚动条的显示样式

此实例主要在选择器:start 和:end 以及伪元素::-webkit-scrollbar、::-webkit- scrollbar-track、::-webkit-scrollbar-thumb 中设置 CSS 样式,从而实现改变拥有 overflow 属性的元素的默认滚动条样式。当在 Google Chrome 浏览器中显示该页面时,上面显示的蓝色圆角滚动条是定制的滚动条,该滚动条的滑块上面部分的轨道呈现绿色,下面部分的轨道呈现红色;下面显示的灰色滚动条是默认的滚动条,如图 504-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .myScrollbar,textarea{width:250px;height:100px; }
  .myScrollbar::-webkit-scrollbar { width: 16px;}
```

```
/ * 设置垂直滚动条的上半边或水平滚动条的左半边背景为绿色 * /
.myScrollbar:: - webkit - scrollbar - track - piece:start {background - color: green;}
/ * 设置垂直滚动条的下半边或水平滚动条的右半边背景为红色 * /
.myScrollbar:: - webkit - scrollbar - track - piece:end {background - color: red;}
/ * 定义滚动条的轨道,内阴影 * /
.myScrollbar:: - webkit - scrollbar - track { - webkit - box - shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
border - radius: 0px; background - color: lightgray; }
/ * 定义滑块,内阴影及圆角 * /
.myScrollbar:: - webkit - scrollbar - thumb {border - radius: 10px; background - color: deepskyblue;
- webkit - box - shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);}
</style></head>
<body><center><form><p>显示定制的滚动条:<textarea class = "myScrollbar">赤壁之战,是指孙权、刘备联军于建安十三年(208 年)在长江赤壁(今湖北省赤壁市西北)一带大破八十万曹军,奠定三国三足鼎立基础的以少胜多,以弱胜强的著名战役。这是中国历史上以弱胜强,以少胜多的著名战争之一,也是三国时期"三大战役"中最为著名的一场。它也是中国历史上第一次在长江流域进行的大规模江河作战,标志着中国军事政治中心不再限于黄河流域。孙刘联军以火攻大破曹军,曹操北回,孙、刘各自夺去荆州的一部分。</textarea></p>
<p>显示默认的滚动条:<textarea>赤壁之战,是指孙权、刘备联军于建安十三年(208 年)在长江赤壁(今湖北省赤壁市西北)一带大破八十万曹军 ,奠定三国三足鼎立基础的以少胜多,以弱胜强的著名战役。这是中国历史上以弱胜强,以少胜多的著名战争之一,也是三国时期"三大战役"中最为著名的一场。它也是中国历史上第一次在长江流域进行的大规模江河作战,标志着中国军事政治中心不再限于黄河流域。孙刘联军以火攻大破曹军,曹操北回,孙、刘各自夺去荆州的一部分。</textarea></p></form></center></body></html>
```



图　504-1

上面有底纹的代码是此实例的核心代码。在该部分代码中,::-webkit-scrollbar- track、::-webkit-scrollbar-thumb、::-webkit-scrollbar 是在 Google Chrome 浏览器中被支持的滚动条样式定制的 3 个伪元素,:start 和:end 是支持 Google Chrome 浏览器的滚动条样式定制的两个选择器,这些伪元素和选择器支持拥有 overflow 属性的区域、列表框、下拉菜单、textarea 等元素的滚动条自定义样式。大多数选择器都可以应用到伪元素中,常用的选择器说明如下。

(1) horizontal:horizontal 选择器适用于任何水平方向的滚动条。

(2) vertical:vertical 选择器适用于任何垂直方向的滚动条。

(3) decrement:decrement 选择器适用于表示递减的按钮或轨道碎片,例如可以使区域向上或者向右移动的区域和按钮。

(4) increment:increment 选择器适用于表示递增的按钮或轨道碎片,例如可以使区域向下或者向左移动的区域和按钮。

（5）start：start 选择器适用于按钮和轨道碎片，表示对象是否放在滑块的前面。

（6）end：end 选择器适用于按钮和轨道碎片，表示对象是否放在滑块的后面。

（7）double-button：double-button 选择器适用于按钮和轨道碎片，判断轨道结束的位置是否为一对按钮，也就是轨道碎片紧挨着一对在一起的按钮。

（8）single-button：single-button 选择器适用于按钮和轨道碎片，判断轨道结束的位置是否为一个按钮，也就是轨道碎片紧挨着一个单独的按钮。

（9）no-button：no-button 选择器表示轨道结束的位置没有按钮。

（10）corner-present：corner-present 选择器表示滚动条的角落是否存在。

（11）window-inactive：window-inactive 选择器适用于所有滚动条，表示包含滚动条的区域，焦点不在该窗口的时候。

此实例的源文件名是 myHtmlB172.html。

505　使用伪元素定制密钥对生成器字段的样式

此实例主要通过在伪元素::-webkit-keygen-select 中设置 CSS 样式实现改变密钥对生成器字段的默认样式。当在 Google Chrome 浏览器中显示该页面时，“定制的密钥对生成器字段:”展开后的效果如图 505-1 所示，“默认的密钥对生成器字段:”展开后的效果如图 505-2 所示。有关此实例的主要代码如下。

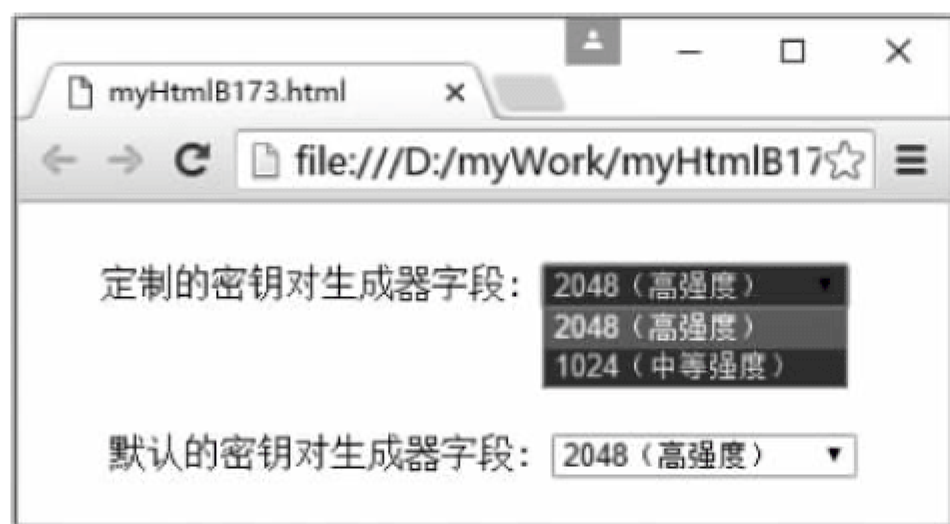


图　505-1



图　505-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .myKeygen{}
  .myKeygen::-webkit-keygen-select{ border: 1px solid #A0B3D6;
          background-color:blue; color: yellow; }
 </style></head>
<body><center><form  action="/" method="get">
 <br>定制的密钥对生成器字段：<keygen name="securityCustom" class="myKeygen" />
  <br><br><br><br>默认的密钥对生成器字段：<keygen name="securityDefault" />
 </form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-keygen-select 是在 Google Chrome 浏览器中被支持的改变密钥对生成器字段的伪元素，如果需要定制密钥对生成器字段（元素）的显示样式，只需要在该伪元素中添加 CSS 代码即可。

此实例的源文件名是 myHtmlB173.html。

506 使用伪元素自定义 meter 度量衡的样式

此实例主要在伪元素::-webkit-meter-bar、::-webkit-meter-optimum-value、::- webkit-meter-suboptimum-value、::-webkit-meter-even-less-good-value 中设置 CSS 样式，从而实现改变 meter 元素的默认样式。当在 Google Chrome 浏览器中显示该页面时，"默认的 meter 元素："和"定制的 meter 元素："的显示效果如图 506-1 所示。有关此实例的主要代码如下。



图 506-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  .myMeter { }
  .myMeter::-webkit-meter-bar { height: 1em; background: gray;
                                border: 1px solid black; }
  .myMeter::-webkit-meter-optimum-value { background: green;}
  .myMeter::-webkit-meter-suboptimum-value { background: yellow; }
  .myMeter::-webkit-meter-even-less-good-value { background: red;}
 </style></head>
<body><center><form action="/" method="get">
 <p>默认的 meter 元素:
   <meter low="69" high="80" max="100" optimum="100" value="100">优</meter>
   <meter low="69" high="80" max="100" optimum="100" value="70">良</meter>
   <meter low="69" high="80" max="100" optimum="100" value="10">差</meter></p>
 <p>定制的 meter 元素:
   <meter class="myMeter" low="69" high="80" max="100" optimum="100" value="100">优</meter>
   <meter class="myMeter" low="69" high="80" max="100" optimum="100" value="70">良</meter>
   <meter class="myMeter" low="69" high="80" max="100" optimum="100" value="10">差</meter></p>
</form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-meter-bar 伪元素用于设置 meter 元素 3 个 bar 的共同样式，::-webkit-meter-optimum-value 伪元素用于设置 meter 元素"优"bar 的样式，::-webkit-meter-suboptimum-value 伪元素用于设置 meter 元素"良"bar 的样式，::-webkit-meter-even-less-good-value 伪元素用于设置 meter 元素"差"bar 的样式。

此实例的源文件名是 myHtmlB174.html。

507 使用伪元素重置 placeholder 占位符

此实例主要通过设置伪元素::-webkit-input-placeholder 实现重置输入框的 placeholder 占位符的样式。当在 Google Chrome 浏览器中显示该页面时，在输入框中显示的浅蓝色文本"输入专业首字试试"就是重置之后的样式，如图 507-1 所示，默认该文本的颜色是暗灰色。有关此实例的主要代码如下。

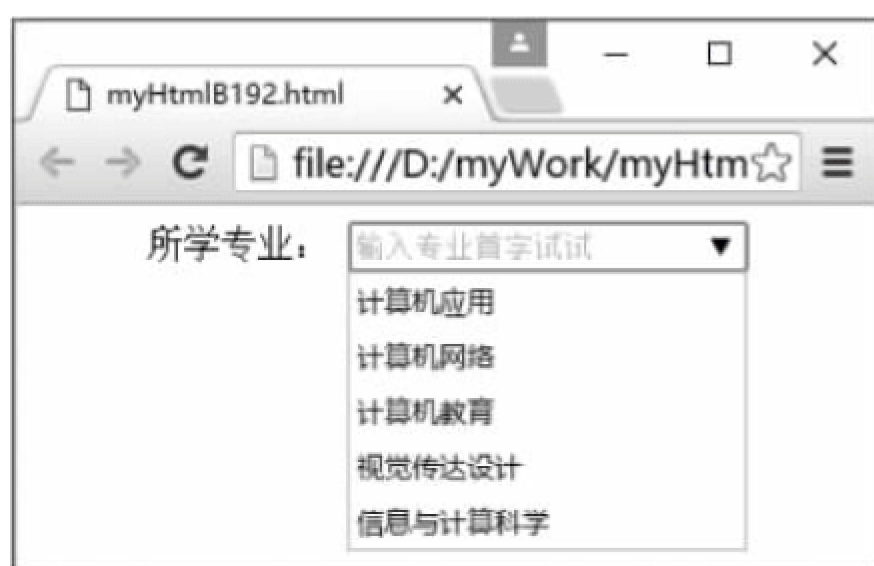


图 507-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /* 设置 placeholder 占位符文本颜色为浅蓝色 */
  ::-webkit-input-placeholder { color: lightblue;}
 </style></head>
<body><section align="center">所学专业:
 <input id="myMajor" list="myData" placeholder="输入专业首字试试"/>
  <datalist id="myData">
   <option value="计算机应用"><option value="计算机网络">
   <option value="计算机教育"><option value="视觉传达设计">
   <option value="信息与计算科学"></datalist></section></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，placeholder 是 HTML5 中 input 元素的新属性，英文意思是占位符，它一般表示 input 输入框的默认提示值。如果需要修改该属性的默认样式，应该在伪元素::-webkit-input-placeholder 中设置。在伪元素::-webkit-input-placeholder 中可以设置几个和文本有关系的 CSS 样式，例如 color、font-style、font-variant、background、text-decoration 等。

此实例的源文件名是 myHtmlB192.html。

508 使用伪元素隐藏搜索框右侧的取消按钮

此实例主要在伪元素::-webkit-search-cancel-button 中设置 display 属性为 none，从而实现隐藏搜索框右侧的取消按钮。当在 Google Chrome 浏览器中显示该页面时，如果在上面的搜索框中输入内容，则不显示取消按钮"×"；如果在下面的搜索框中输入内容，则将显示默认的取消按钮"×"，单击取消按钮"×"，将清除刚才输入的内容，如图 508-1 所示。有关此实例的主要代码如下。

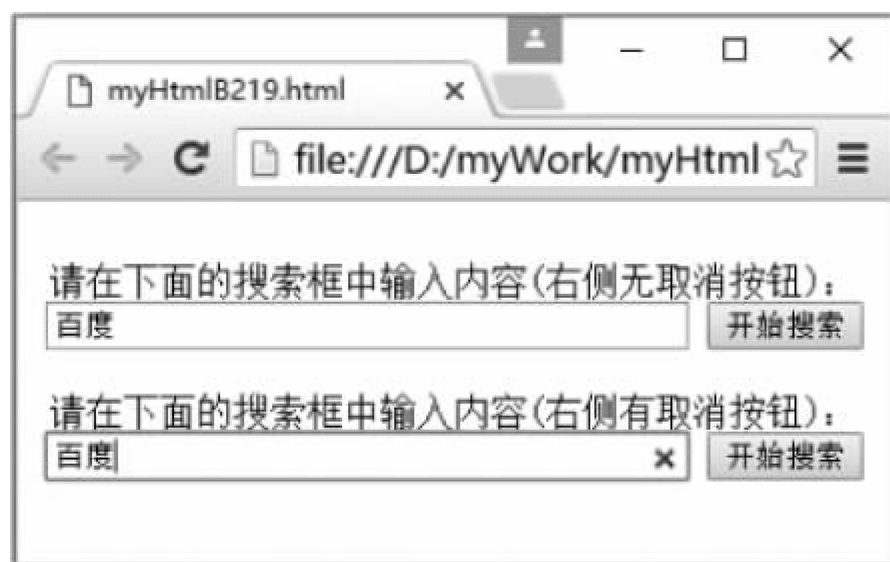


图 508-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  #myCancel::-webkit-search-cancel-button{display: none;}
  input{width:280px; }
 </style></head>
<body><center><form><br>
  <label>请在下面的搜索框中输入内容(右侧无取消按钮): </label><br>
  <input id="myCancel" type="search"/>
  <button type="submit">开始搜索</button></form><br/>
 <form><label>请在下面的搜索框中输入内容(右侧有取消按钮): </label><br>
  <input id="myNormal" type="search"/>
  <button type="submit">开始搜索</button></form></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,::-webkit-search-cancel- button 伪元素用于管理搜索框的取消按钮;display:none 表示隐藏该取消按钮。

此实例的源文件名是 myHtmlB219.html。

509 使用伪元素使 range 滑槽呈现渐变色

此实例主要在伪元素::-webkit-slider-runnable-track、::-webkit-slider-thumb 中设置 CSS 样式,从而实现 range 滑槽根据当前位置呈现渐变色风格。当在 Google Chrome 浏览器中显示该页面时,拖动新定制 range 的红色圆形滑块向右滑动,则红色圆形滑块左侧的滑槽将呈现浅蓝色的渐变色,并且滑槽的渐变色变化随着红色圆形滑块的位置变化而变化,当然下面的图像透明度也随着红色圆形滑块的位置变化而变化,如图 509-1 所示。有关此实例的主要代码如下。



图 509-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $.fn.RangeSlider = function(myPara) {
```

```
      this.mySlider = {
       min: myPara && !isNaN(parseFloat(myPara.min)) ? Number(myPara.min) : null,
       max: myPara && !isNaN(parseFloat(myPara.max)) ? Number(myPara.max) : null,
       step: myPara && Number(myPara.step) ? myPara.step : 1,
       callback: myPara && myPara.callback ? myPara.callback : null };
      var $myRange = $(this); var min = this.mySlider.min;
      var max = this.mySlider.max; var step = this.mySlider.step;
      var callback = this.mySlider.callback;
      $myRange.attr('min', min).attr('max', max).attr('step', step);
      $myRange.bind("input", function(e) {
      $myRange.attr('value', this.value);
      //根据当前值创建线性渐变的背景
      $myRange.css('background', 'linear-gradient(to right, #059CFA, white '+ this.value + '%, white)');
      if ($.isFunction(callback)) { callback(this); } });};
   var change = function ($myRange) {
     $('#myCurrentValue').text('滑块当前(图像透明度的百分比值)位置: '+ $myRange.value);
    //根据滑块值重置图像的透明度
     $("img").css("opacity", (100 - $myRange.value)/ 100);
   }
   $('input').RangeSlider({min: 0, max: 100, step: 0.1, callback: change});});
 </script>
 <style type = "text/css">
   input[type = range] { -webkit-appearance: none; width: 400px; border-radius: 10px; }
   input[type = range]::-webkit-slider-thumb { -webkit-appearance: none; }
    /*定制滑槽*/
    input[type = range]::-webkit-slider-runnable-track {height: 15px;border-radius: 10px; box-
 shadow: 0 1px 1px #DEF3F8, inset 0 0.125em 0.125em #0D1112;}
    input[type = range]:focus { outline: none;}
    /*定制滑块(红色的圆饼)*/
    input[type = range]::-webkit-slider-thumb { -webkit-appearance: none; height: 25px; width: 25px;
 margin-top: -5px; background: red; border-radius: 50%;border: solid 0.125em rgba(205, 224, 230,
 0.5); box-shadow: 0 0.125em 0.125em #3B4547; }
    img { width:400px; height:250px; border-radius: 5px; margin: 10px;}
   </style></head>
 <body><center><p id = "myCurrentValue">滑块当前(图像透明度的百分比值)位置: </p>
   <div><input type = "range" value = "0"></div>
   <img src = "img/B245.jpg"></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，::-webkit-slider- runnable-track 伪元素的 CSS 代码用于设置 range 元素的滑槽样式；::-webkit-slider- thumb 伪元素的 CSS 代码用于设置 range 元素的滑块样式；-webkit-appearance：none 用于去除系统默认 appearance 的样式，此实例即是取消 range 元素的默认样式；$myRange. css('background'，'linear-gradient(to right，#059CFA，white '+this. value+'%，white)')用于根据当前滑块位置在滑块左侧生成渐变色的滑槽。

此实例的源文件名是 myHtmlB245. html。

510 使用伪元素自定义渐变色风格的滚动条

此实例主要在伪元素::-webkit-scrollbar-track、::-webkit-scrollbar-thumb、:: -webkit-scrollbar 中设置 CSS 样式，从而创建自定义渐变色风格的滚动条。当在 Google Chrome 浏览器中显示该页面

时，文字的右侧将显示红绿渐变的滚动条，如图 510-1 所示；拖动红绿渐变的滚动条文字即向下滚动，如图 510-2 所示。有关此实例的主要代码如下。

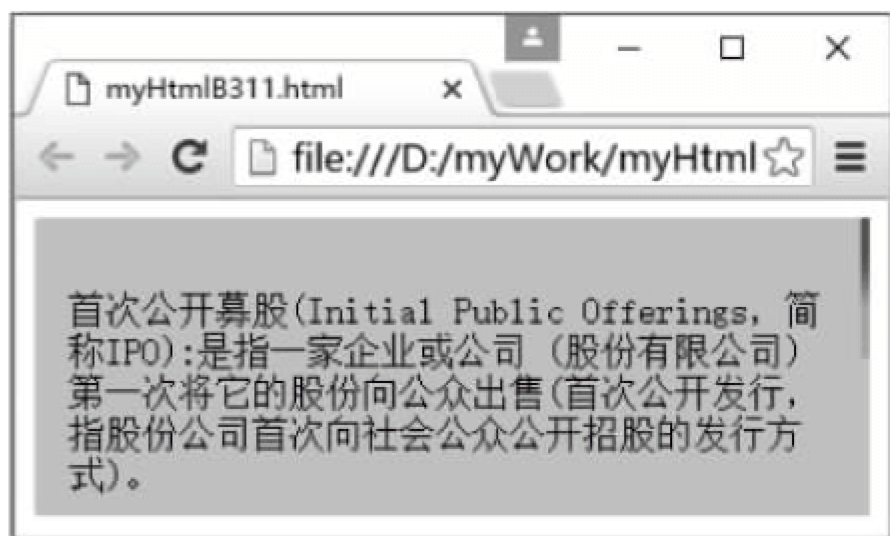


图 510-1

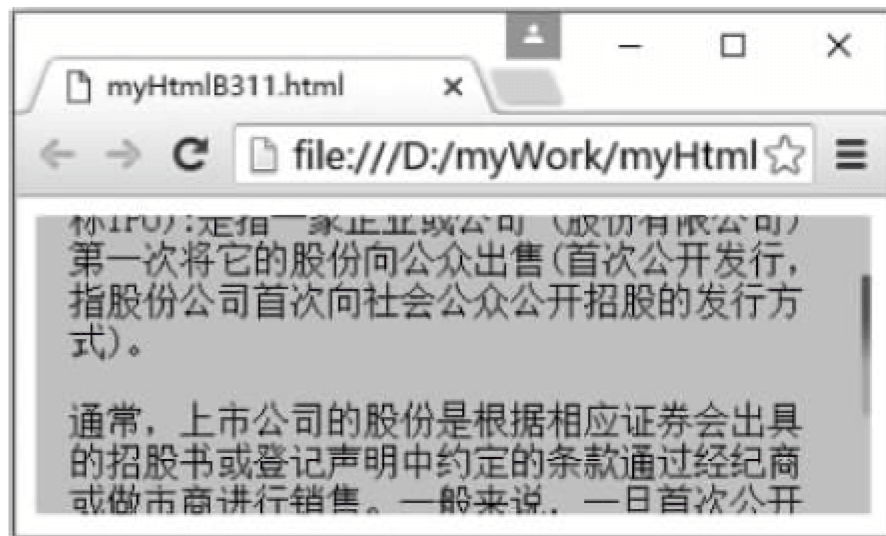


图 510-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myScrollbar::-webkit-scrollbar-track { -webkit-box-shadow: inset 0 0 0 rgba(0, 0, 0, 0.8);
background-color:lightgrey; }
  .myScrollbar::-webkit-scrollbar { width: 4px; } /* 设置滚动条的宽度 */
  /* 设置滚动条的渐变色滑块 */
  .myScrollbar::-webkit-scrollbar-thumb {background-image: -webkit-gradient(linear, right
bottom, left top, color-stop(0.30, lightgreen), color-stop(0.90, red)); }
  article { padding: 15px; overflow-y: scroll; height: 100px; background: lightgrey;
            border-radius: 2px; }
 </style></head>
<body><article class="myScrollbar"><p>首次公开募股(Initial Public Offerings,简称 IPO):是指一家企
业或公司(股份有限公司)第一次将它的股份向公众出售(首次公开发行,指股份公司首次向社会公众公开招股
的发行方式)。</p>
    <p>通常,上市公司的股份是根据相应证券会出具的招股书或登记声明中约定的条款通过经纪商或做市商
进行销售。一般来说,一旦首次公开上市完成后,这家公司就可以申请到证券交易所或报价系统挂牌交易。有
限责任公司在申请 IPO 之前,应先变更为股份有限公司。</p></article></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，overflow-y：scroll 表示对内容的上/下边缘提供滚动机制；height：100px 表示可视高度是 100px。当需要元素支持滚动功能时通常应该设置上述两个属性。.myScrollbar::-webkit-scrollbar-track { }用于自定义滚动条的轨道（里面装有 thumb）样式；.myScrollbar::-webkit-scrollbar { }用于自定义滚动条的整体部分；.myScrollbar::-webkit-scrollbar-thumb { }用于自定义滚动条里面的小方块，能上下左右移动（取决于是垂直滚动条还是水平滚动条）。

此实例的源文件名是 myHtmlB311.html。

511 使用伪元素实现菱形滑块的 range 元素

此实例主要在伪元素::-webkit-slider-thumb 中使用菱形的 png 图像设置 range 元素的滑块，从而实现改变 range 元素默认的矩形风格的滑块样式。当在 Google Chrome 浏览器中显示该页面时，拖动菱形滑块即可改变 range 元素的大小，效果分别如图 511-1 和图 511-2 所示。有关此实例的主要代码如下。

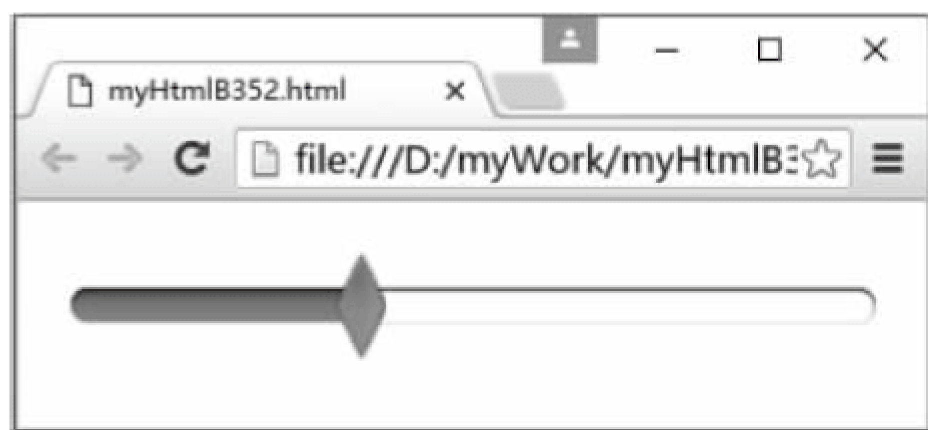


图 511-1

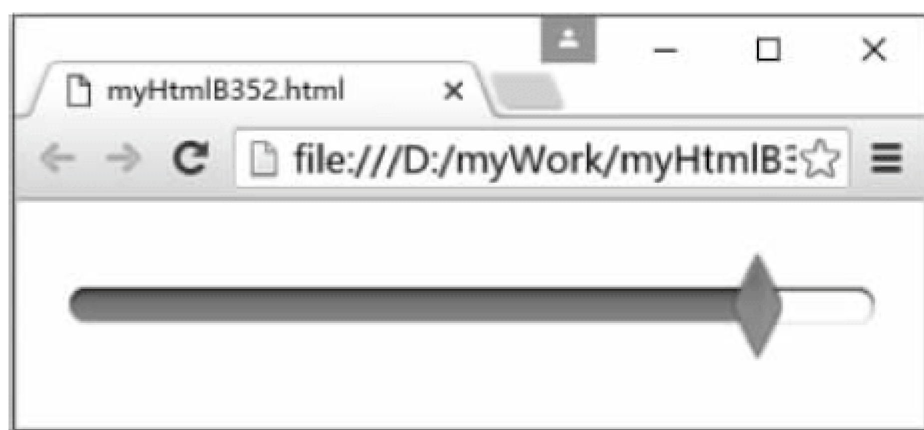


图 511-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function(){//响应鼠标拖动滑块事件
   $("input[type=range]").on("propertychange input",function(){
    $("input[type=range]").css( 'background-size', $(this).val() +'% 100%' );
  }) });
 </script>
 <style type="text/css">
  /*设置 range 的基本样式*/
  .myRange{ -webkit-appearance: none; width: 350px; border-radius: 10px; background: -webkit-
linear-gradient(green,lime) no-repeat; background-size: 0% 100%; margin: 30px auto; }
  /*使用菱形的 png 图片设置 range 的滑块*/
  .myRange::-webkit-slider-thumb { -webkit-appearance: none; height: 47px; width: 23px; margin-
top: -16px; background-repeat:no-repeat; background-image: url(img/B352.png); }
  /*设置 range 滑槽的基本样式*/
  .myRange::-webkit-slider-runnable-track{height: 15px; border-radius: 15px; box-shadow: 0 1px
1px #DEF3F8, inset 0 0.125em 0.125em #0D1112; }
  .myRange:focus { outline: none; }  /*隐藏焦点状态的 range 轮廓线*/
 </style></head>
<body><div align="center"><input type="range" max="100" min="0" value="0" class="myRange"/>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,::-webkit-slider- runnable-track 伪元素的 CSS 代码用于设置 range 元素的滑槽样式;::-webkit-slider- thumb 伪元素的 CSS 代码用于设置 range 元素的滑块样式,菱形滑块即是在此伪元素中通过设置背景实现的。如果需要在 Google Chrome 浏览器中定制 range 元素的样式,直接在其中添加 CSS 代码即可。

此实例的源文件名是 myHtmlB352.html。

512 通过 for 属性自定义默认的复选框图标

此实例主要设置 label 元素的背景图像为自定义复选框图标,并通过 for 属性将其与 input 元素关联起来,从而实现自定义默认的复选框图标。当在 Google Chrome 浏览器中显示该页面时,自定义复选框图标的未选中状态如图 512-1 所示;单击之,则显示自定义复选框图标的选中状态,如图 512-2 所示;再单击之,则显示自定义复选框图标的未选中状态,如图 512-1 所示。有关此实例的主要代码如下。



图 512-1



图 512-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
   $(function() {
    $('#myCheckbox').change(function() {//获取复选框的 checked 属性值
     var check = $('#myCheckbox').prop('checked');
     if (check) {//显示提示信息
      $('#myStatus').text('【已选中】');
     } else {
      $('#myStatus').text('【未选中】');   } });
   //自动触发 change 事件
   $('#myCheckbox').trigger('change'); });
 </script>
 <style>
   #myCheckbox { display: none; }
   #myStatus, #myInfo { display: inline; }
   label { display: inline-block; width: 20px;   height: 20px; background-image: url(img/B234A.png);
background-size: 100% 100%; }
   #myCheckbox:checked+label { background-image: url(img/B234B.png);}
 </style></head>
<body><center>
 <input type="checkbox" id="myCheckbox"/><label for="myCheckbox"></label>
 <p id="myInfo">C#高级编程</p><p id="myStatus"></p></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，#myCheckbox{ display：none;}表示隐藏复选框；< label for="myCheckbox">表示将 label 元素与隐藏复选框关联起来，因此单击 label 元素即是单击隐藏复选框。在 CSS 中，label 元素可用于为 input 元素服务，为其定义标记，其 for 属性规定 label 与哪个表单元素绑定。label 和表单控件绑定的方式有以下两种。

（1）将表单控件作为 label 的内容，这就是隐式绑定，此时不需要 for 属性，绑定的控件也不需要 id 属性。

（2）为 label 元素的 for 属性命名一个目标表单的 id，这就是显式绑定，例如此实例中的< label for="myCheckbox">。

此实例的源文件名是 myHtmlB234.html。

513 通过设置五星字符的颜色实现星级评分

此实例主要通过检测鼠标指针悬浮或单击五星字符事件并据此设置五星字符的颜色来实现星级评分。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第 3 颗星上，则会显示浮动框提示信息，例如“一般”，如图 513-1 所示，单击则会保存选择结果。有关此实例的主要代码如下。



图 513-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script type="text/javascript">
  window.onload = function() {
   var star = document.getElementById("star");
   var myStar = star.getElementsByTagName("li");
   var myInfo = document.getElementById("myInfo");
   var result = document.getElementById("result");
   var i = 0; var j = 0; var len = myStar.length;
   var word = ['很差', '差', '一般', "好", "很好"]
   for (i = 0; i < len; i++) {
    myStar[i].index = i;
    myStar[i].onmouseover = function() {       //响应鼠标指针悬浮于某个五星
     for (i = 0; i < len; i++) {               //设置所有五星为未选中颜色
      myStar[i].className = "origin";
     }
    myInfo.style.display = "block";
    myInfo.innerHTML = word[this.index];
    for (i = 0; i <= this.index; i++) {        //设置当前五星至开始的所有五星为选中颜色
      myStar[i].className = "act";
    } }
   myStar[i].onmouseleave = function () {      //响应鼠标指针离开某个五星
    for (i = 0; i < len; i++) {                //设置所有五星为未选中颜色
     myStar[i].className = "origin";
    }
    myInfo.style.display = "none";
    for (i = 0; i <= this.index; i++) {        //设置当前五星至开始的所有五星为选中颜色
     myStar[i].className = "act";
  } }
  myStar[i].onclick = function () {            //响应鼠标单击某个五星
   result.innerHTML = (this.index + 1) + "分";
   for (i = 0; i < this.index; i++) {          //设置当前五星至开始的所有五星为选中颜色
     myStar[i].className = "act"; } } } }
 </script>
 <style type="text/css">
  .myBox { width: 300px; margin: 10px auto; font: 14px/1.5 arial; }
  #star { overflow: hidden; }
  #star li { float: left; width: 40px; height: 40px; margin: 2px; display: inline; color: #999; font:
bold 36px arial; cursor: pointer; }
  #star .act { color: red; }
  #star .origin { color: gray; }
```

```
    #myInfo { width: 80px;  height: 30px; line-height: 30px; border: 1px solid #CCC;
          margin: 10px; text-align: center;  display: none; }
  </style></head>
<body><center><div class="myBox"> 打分结果:
  <span id="result"></span>
   <ul id="star"><li>★</li><li>★</li><li>★</li><li>★</li><li>★</li></ul>
  <div id="myInfo">一般</div></div></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：inline 表示此元素会被显示为内联元素，元素前后没有换行符；myInfo.style.display＝"block"表示此元素将显示为块级元素，此元素前后会带有换行符；myInfo.style.display＝"none"表示此元素不会被显示；#star .act{color：red;}表示当前被选中(和之前所有)五星以红色显示；#star .origin {color：gray;}表示当前未选中(和之后所有)五星以灰色显示。

此实例的源文件名是 myHtmlB236.html。

514　实现以不同的颜色代表不同的星级评分

此实例主要在鼠标指针悬浮或单击事件中设置星级评分的五星字符颜色和评语，从而实现以不同的颜色代表不同的星级评分效果。当在 Google Chrome 浏览器中显示该页面时，评分栏处于初始状态，如果鼠标指针悬浮在第 1 颗五星上，则会显示评语信息，例如“垃圾，简直就是垃圾！”，并且第 1 颗五星呈现红色，如图 514-1 所示，单击则会保存选择结果，如图 514-2 所示；单击“重新进行评分”按钮，则评分栏处于初始状态，如果鼠标指针悬浮在第 4 颗五星上，则会显示评语信息，例如“不错，还可以”，并且前 4 颗五星呈现蓝色，如图 514-3 所示，单击则会保存选择结果，如图 514-4 所示。其他五星具有类似的功能。有关此实例的主要代码如下。



图　514-1



图　514-2



图 514-3



图 514-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   for (var i=1; i<=5; i++) {                          //响应鼠标指针悬浮于五星
     $(".myStar" + i).hover(function() {
      var index = $(this).attr("class").substr(6, 1);
      for (var i = index; i>0; i--) {                  //设置五星颜色
        $(".myStar" + i).addClass("c" + index);
      }
      switch (index) {                                 //设置五星评语
       case "1":
              $(".myInfo").addClass("c1").text("垃圾,简直就是垃圾!");
              break;
       case "2":
              $(".myInfo").addClass("c2").text("太糟糕了!"); break;
       case "3":
              $(".myInfo").addClass("c3").text("一般般吧!");break;
       case "4":
              $(".myInfo").addClass("c4").text("不错,还可以");break;
       case "5":
              $(".myInfo").addClass("c5").text("Good,一次愉快的购物体验!");
              break;
     }  }, function() {                                //移除五星颜色和评语设置
       var index = $(this).attr("class").substr(6, 1);
       for (var i = index; i>0; i--) { $(".myStar" + i).removeClass("c" + index); }
        $(".myInfo").removeClass("c" + index).text("");
   });
    $(".myStar" + i).bind("click", function() {        //响应鼠标单击某个五星
```

```
        var index = $ (this).attr("class").substr(6, 1);
        for (var i = 1; i <= index; i++) { $ (".myStar" + i).unbind(); }
        while (++ index <= 5) {                    //评分之后清除无用的五星
         $ (".myStar" + index).css("display", "none");  } });}
      $ ("#myBtnMark").click(function() {          //重新进行评分
       location.reload();  });});
   </script>
   <style type = "text/css">
     * { margin: 0; }
    .myBox { margin - top: 15px; width: 500px; }
    .list { margin: 20px 10px; width: 200px;  float: left; }
    a { font - size: 25px; text - decoration: none; color: lightgray; }
    span { font - size: 20px;line - height: 70px; }
    /* 设置每档次颜色 */
    .c1 { color: red; }  .c2 { color: pink; }
    .c3 { color: cyan; }  .c4 { color: blue; }
    .c5 { color: green; }
    button { margin - top: 20px; font - size: 20px;  width:420px; }
   </style></head>
  <body>
  <div align = "center">
   <button id = "myBtnMark">重新进行评分</button>
    <div class = "myBox"> <div class = "list">
      <a href = "#" class = "myStar1">★</a> <a href = "#" class = "myStar2">★</a>
      <a href = "#" class = "myStar3">★</a><a href = "#" class = "myStar4">★</a>
      <a href = "#" class = "myStar5">★</a></div>
     <span class = "myInfo">亲,请客观给出评价</span></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$(". myStar"+i). addClass("c"+index)用于设置五星颜色,addClass()方法是 jQuery 的 CSS 属性操作方法,它用于向被选元素添加一个或多个类,该方法不会移除已存在的 class 属性,仅仅添加一个或多个 class 属性;$(". myStar"+i). removeClass("c"+index)用于移除五星颜色,removeClass()方法是 jQuery 的 CSS 属性操作方法,它用于从被选元素移除一个或多个类,如果没有规定参数,则该方法将从被选元素中删除所有类。

此实例的源文件名是 myHtmlB341. html。

515　自定义光标模拟粉笔在黑板上涂鸦

此实例主要为画布标签添加 mousedown、mousemove、mouseup 等事件响应方法,并使用 HTML5 的图形绘制方法 fillRect()、clearRect()、moveTo()、lineTo()等,从而模拟自定义粉笔光标在黑板上涂鸦。当在 Google Chrome 浏览器中显示该页面时,在黑色的自定义画布上即可通过移动鼠标进行文字涂鸦,如图 515-1 所示(由于粉笔是光标,截图没有截下来)。有关此实例的主要代码如下。

```
  <!doctype html><html><head><meta charset = "UTF - 8">
   <script src = "js/jquery - 3.1.1.min.js"></script><script>
     $ (function(){
      var lastX, lastY, myDiameter = 7;
      var myContext = $ ("#myCanvas")[0].getContext("2d");
```

```
    myContext.fillStyle = "black";                              //绘制黑色背景矩形
    myContext.fillRect(0,0,400,250);
    var isdraw = false;
    $("#myCanvas").on("mousedown",function(e){                  //开始绘制
     isdraw = true; lastX = e.offsetX;  lastY = e.offsetY;   });
    $("#myCanvas").on("mousemove",function(e){                   //执行绘制
     $(this).css("cursor","url('img/A180myPen.png'),auto");     //设置粉笔光标
     if(isdraw){
       myContext.beginPath();
       myContext.moveTo(lastX,lastY);
       myContext.lineTo(e.offsetX, e.offsetY);
       myContext.stroke();
       var length = Math.round(Math.sqrt(Math.pow(e.offsetX - lastX,2) + Math.pow (e.offsetY - lastY,2))/
(5/myDiameter));
       var xUnit = (e.offsetX - lastX)/length;
       var yUnit = (e.offsetY - lastY)/length;
       for(var i = 0; i < length; i++){
         var xCurrent = lastX + (i * xUnit);
         var yCurrent = lastY + (i * yUnit);
         var xRandom = xCurrent + (Math.random() - 0.5) * myDiameter * 1.2;
         var yRandom = yCurrent + (Math.random() - 0.5) * myDiameter * 1.2;
         myContext.clearRect(xRandom, yRandom, Math.random() * 2 + 2, Math.random() + 1);
       }
       lastX = e.offsetX; lastY = e.offsetY;
     }});
     $("#myCanvas").on("mouseup",function(){                     //结束绘制
      isdraw = false;  });});
 </script>
 <style type = "text/css">
  canvas{ border - radius: 5px;}
 </style></head>
<body><div align = "center"><canvas id = "myCanvas" width = "400px" height = "250px"> </canvas></div>
</body></html>
```



图 515-1

上面有底纹的代码是此实例的核心代码。在该部分代码中，$(this).css("cursor", "url('img/A180myPen.png'),auto")用于设置在使用鼠标涂鸦时鼠标光标呈现粉笔状态，A180myPen.png 就

是一幅粉笔图像；fillRect()方法用于绘制“已填色”的矩形，默认的填充颜色是黑色，可以使用 fillStyle 属性来设置填充绘图的颜色、渐变或模式；clearRect()方法用于清空参数指定的矩形内的像素；moveTo()方法用于把路径移动到画布中的指定点，但是不创建线条；lineTo()方法用于添加一个新点，然后在画布中创建从该点到最后指定点的线条。

此实例的源文件名是 myHtmlA180.html。

516　允许用户使用拖动方式重置元素的尺寸

此实例主要通过设置元素的 resize 属性实现允许用户以拖动的方式重置元素的尺寸。当在 Google Chrome 浏览器中显示该页面时，左、右两个文本块(元素)显示的文本如图 516-1 所示，分别拖动左、右两个文本块右下角的拖动控制块则会改变文本块(元素)的尺寸，出现如图 516-2 所示的文本对齐显示效果。有关此实例的主要代码如下。



图　516-1



图　516-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myContainer {display: -webkit-flex;}
  #myLeft { width: 200px;padding: 15px;background-color: lightcyan;
            overflow: auto; resize:both; }
  #myRight { width: 200px; padding: 15px; background-color: lightgreen;
            overflow: auto; resize:both; }
  #myLeft,#myRight {box-sizing: border-box;}
  h3 { text-align: center; }
 </style></head>
<body><div id="myContainer"><div id="myLeft">
  <h3>行行重行行</h3>
  <p>行行重行行,与君生别离。相去万余里,各在天一涯。
     道路阻且长,会面安可知?胡马依北风,越鸟巢南枝。
     相去日已远,衣带日已缓。浮云蔽白日,游子不顾返。
     思君令人老,岁月忽已晚。弃捐勿复道,努力加餐饭。</p></div>
 <div id="myRight">
  <h3>西北有高楼</h3>
  <p>西北有高楼,上与浮云齐。交疏结绮窗,阿阁三重阶。
```

```
        上有弦歌声,音响一何悲!谁能为此曲,无乃杞梁妻。
        清商随风发,中曲正徘徊。一弹再三叹,慷慨有余哀。
        不惜歌者苦,但伤知音稀。愿为双鸿鹄,奋翅起高飞。</p></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,resize 属性规定是否可由用户调整元素的尺寸,如果希望此属性生效,需要同时设置元素的 overflow 属性,overflow 属性的值可以是 auto、hidden、scroll。resize 属性的语法格式如下。

```
resize: none|both|horizontal|vertical;
```

其中,各属性值的意义如表 516-1 所示。

表 516-1　resize 属性的值及意义

值	说　明
none	用户无法调整元素的尺寸
both	用户可调整元素的高度和宽度
horizontal	用户可调整元素的宽度
vertical	用户可调整元素的高度

此实例的源文件名是 myHtmlB108.html。

517　创建距离活动结束的剩余时间倒计时牌

此实例主要通过在定时器函数 setInterval()中不断递减时间值,从而创建距离活动结束剩余时间的倒计时牌。当在 Google Chrome 浏览器中显示该页面时,倒计时牌将每秒刷新一次剩余时间,如图 517-1 所示。有关此实例的主要代码如下。

图　517-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  var myDuration = parseInt(60 * 60);                //倒计时总秒数为 3600,即 1 小时
  function timer(myDuration) {
   window.setInterval(function(){                    //每隔 1 秒刷新 1 次
   var day = 0, hour = 0, minute = 0, second = 0;    //时间默认值
   if (myDuration > 0) {                             //数字取整
    day = Math.floor(myDuration/(60 * 60 * 24));
    hour = Math.floor(myDuration/(60 * 60)) - (day * 24);
```

```
        minute = Math.floor(myDuration/60) - (day * 24 * 60) - (hour * 60);
        second = Math.floor(myDuration) - (day * 24 * 60 * 60) - (hour * 60 * 60) - (minute * 60);
      }
      if (minute <= 9) minute = '0' + minute;
      if (second <= 9) second = '0' + second;
      $('#myDay').html(day + "天");
      $('#myHour').html('<s id="h"></s>' + hour + '时');
      $('#myMinute').html('<s></s>' + minute + '分');
      $('#mySecond').html('<s></s>' + second + '秒');
      myDuration--;
    }, 1000); }
   $(function() { timer(myDuration); });
  </script>
  <style type="text/css">
   /* 设置显示板的基本样式 */
   .myItem strong { background: red; color: white; line-height: 49px; font-size: 30px;
                   font-family: Arial;padding: 0 10px; margin-right: 10px; border-radius: 5px; box-
                   shadow: 2px 2px 5px rgba(0, 0, 0, 0.2); }
   .myItem { font-size: 25px;font-weight: bold;}
   #myDay { line-height: 49px;color: red; font-size: 32px; margin: 0 10px;
            font-family: Arial, Helvetica, sans-serif;}
  </style></head>
 <body><div class="myItem" align="center"><p>距离本场活动结束还剩: </p>
   <span id="myDay">0 天</span><strong id="myHour">0 时</strong>
   <strong id="myMinute">0 分</strong><strong id="mySecond">0 秒</strong></div>
 </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,Math.floor()是 JavaScript 方法,它用于舍去数字的小数部分,仅保留整数。另外还有两个类似的方法,ceil()和 round(),它们的取整示例如下。

```
Math.ceil(12.2)          //返回 13
Math.ceil(12.7)          //返回 13
Math.floor(12.2)         //返回 12
Math.floor(12.7)         //返回 12
Math.round(12.2)         //返回 12,四舍五入
Math.round(12.7)         //返回 13,四舍五入
```

此实例的源文件名是 myHtmlA179.html。

第6部分 布局

518 动态重置盒布局中各列内容的顺序

此实例主要通过设置列的 order 属性实现动态重置盒布局中各列内容的显示顺序。当在 Google Chrome 浏览器中显示该页面时，单击“按照 231 排列”按钮，则各列的显示顺序如图 518-1 所示；单击“按照 132 排列”按钮，则各列的显示顺序如图 518-2 所示。有关此实例的主要代码如下。



图 518-1



图 518-2

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script><script language = "javascript">
  $(document).ready(function() {
    $("#myBtnOrder").click(function() {          //按照 231 排列
      $("#myLeft").css("order","2"); $("#myMiddle").css("order","3");
      $("#myRight").css("order","1");
    });
    $("#myBtnIndex").click(function() {          //按照 132 排列
      $("#myLeft").css("order","1"); $("#myMiddle").css("order","3");
```

```
        $("#myRight").css("order","2");
  }); });
 </script>
 <style type="text/css">
   #myContainer{ display: -webkit-flex;}
   #myLeft{ width: 200px;padding: 10px;background-color: lightblue;}
   #myMiddle{width: 150px; padding: 10px; background-color: lightpink;}
   #myRight{width: 100px; padding: 10px;background-color: lightgreen;}
   #myLeft, #myMiddle, #myRight{box-sizing: border-box;}
   h3{text-align: center;}
 </style></head>
<body>
<p><input type="button" value="按照 231 排列" id="myBtnOrder" style="width:220px"/>
  <input type="button" value="按照 132 排列" id="myBtnIndex" style="width:220px"/></p><div id=
"myContainer">
 <div id="myLeft"><h3>红藕香残玉簟秋</h3><p>红藕香残玉簟秋。轻解罗裳,独上兰舟。云中谁寄锦书
来,雁字回时,月满西楼。花自飘零水自流。一种相思,两处闲愁。此情无计可消除,才下眉头,却上心头。</p>
</div>
 <div id="myMiddle"><h3>寻寻觅觅</h3><p>寻寻觅觅,冷冷清清,凄凄惨惨戚戚。乍暖还寒时候,最难将
息。三杯两盏淡酒,怎敌他、晚来风急?雁过也,正伤心,却是旧时相识。满地黄花堆积。憔悴损,如今有谁堪摘?
守着窗儿,独自怎生得黑?梧桐更兼细雨,到黄昏、点点滴滴。这次第,怎一个愁字了得!</p></div>
 <div id="myRight"><h3>春晚</h3><p>风住尘香花已尽,日晚倦梳头。物是人非事事休,欲语泪先流。闻说
双溪春尚好,也拟泛轻舟。只恐双溪舴艋舟,载不动许多愁。</p></div> </div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myLeft"). css("order","2")用于设置此列在容器中的显示顺序为 2。在 CSS3 中,order 属性用于设置或检索弹性盒模型对象的子元素出现的顺序,order 定义将会影响 position 值为 static 的元素的层叠级别,数值小的会被数值大的盖住。

此实例的源文件名是 myHtmlB078. html。

519　动态重置盒布局中各列内容的方向

此实例主要通过设置盒子的 flex-direction 属性实现动态重置盒布局中各列内容的显示方向。当在 Google Chrome 浏览器中显示该页面时,单击“纵向排列”按钮,则各列的显示方向如图 519-1 所示;单击“横向排列”按钮,则各列的显示方向如图 519-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnOrder").click(function() {            //纵向排列
     $("#myContainer").css("flex-direction","column");
    });
    $("#myBtnIndex").click(function() {            //横向排列
     $("#myContainer").css("flex-direction","row");  });});
 </script>
 <style type="text/css">
   #myContainer{ display: -webkit-flex;}
   #myLeft{width: 150px;padding: 10px; background-color: lightblue;}
   #myMiddle{width: 150px;padding: 10px;background-color: lightpink;}
   #myRight{width: 150px;padding: 10px;background-color: lightgreen;}
```

```
    #myLeft, #myMiddle, #myRight{box-sizing: border-box;}
    h3{text-align: center;}
  </style></head>
<body>
<p><input type="button" value="纵向排列" id="myBtnOrder" style="width:220px"/>
  <input type="button" value="横向排列" id="myBtnIndex" style="width:220px"/></p>
<div id="myContainer">
 <div id="myLeft"><h3>九日齐山登高</h3><p>江涵秋影雁初飞,与客携壶上翠微。尘世难逢开口笑,菊花须插满头归。</p></div>
 <div id="myMiddle"><h3>锦瑟</h3><p>锦瑟无端五十弦,一弦一柱思华年。庄生晓梦迷蝴蝶,望帝春心托杜鹃。</p></div>
 <div id="myRight"><h3>春晚</h3><p>风住尘香花已尽,日晚倦梳头。物是人非事事休,欲语泪先流。</p>
</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myContainer").css("flex-direction","column")用于设置容器中的所有列以纵向显示。在CSS3中,flex-direction属性通过定义flex容器的主轴方向来决定flex子项在flex容器中的位置,这将决定flex如何进行排列,该属性的反转取值不影响元素的绘制、语音和导航顺序,只改变流动方向。flex-direction属性的语法格式如下。

```
flex-direction: row | row-reverse | column | column-reverse
```

其中,row表示主轴与行内轴方向作为默认的书写模式,即横向从左到右排列(左对齐);row-reverse表示对齐方式与row相反;column表示主轴与块轴方向作为默认的书写模式,即纵向从上往下排列(顶对齐);column-reverse表示对齐方式与column相反。

此实例的源文件名是myHtmlB079.html。



图 519-1



图 519-2

520 使单行文本在垂直方向上位于盒子的正中

此实例主要通过设置文本的行高等于盒子的高度实现使单行文本在垂直方向上位于盒子的正中。当在 Google Chrome 浏览器中显示该页面时，文本“杨柳青青江水平”正好在垂直方向上位于图像的正中，如图 520-1 所示。有关此实例的主要代码如下。



图 520-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div { width:400px; height:250px; border-radius: 1px;
       line-height: 250px; background-image: url(img/B243.jpg);
       background-size: 100% 100%; }
  p { font-size: 36px; color:white;text-shadow:2px 2px 4px #000000; }
   * { margin: 0; overflow: hidden;}
 </style></head>
<body><center><div><p>杨柳青青江水平</p></div></center>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，height 属性用于设置元素的高度；line-height 属性用于设置行间的距离(行高)。由于在默认情况下文本位于行高(垂直方向)的正中，因此当行高等于盒子的高度时文本也自然位于盒子垂直方向的正中。

此实例的源文件名是 myHtmlB243.html。

521 在图像(盒子)的上端、下端居中显示文本

此实例主要通过设置 display、-webkit-box-pack 和-webkit-box-align 等属性实现在图像(盒子)的上端、下端居中显示文本及其他元素。当在 Google Chrome 浏览器中显示该页面时，单击“上端显示标题”按钮，则标题文本“经典桌面”的显示效果如图 521-1 所示；单击“居中显示标题”按钮，则标题文本“经典桌面”的显示效果如图 521-2 所示；单击“下端显示标题”按钮，则标题文本“经典桌面”的显示效果如图 521-3 所示。有关此实例的主要代码如下。



图 521-1



图 521-2



图 521-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnTop").click(function() {            //上端显示标题
    $("div").css("-webkit-box-align", "start");
   });
   $("#myBtnCenter").click(function() {         //居中显示标题
    $("div").css("-webkit-box-align", "center");
   });
   $("#myBtnBottom").click(function() {         //下端显示标题
    $("div").css("-webkit-box-align", "end");  });});
 </script>
 <style type="text/css">
```

```
    div { width: 450px; height: 300px; border - radius: 10px; background: url(img/B193.jpg); box - shadow:
         5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset; display: - webkit - box; - webkit - box
         - pack: center; - webkit - box - align: center; }
    span { color: aqua; font - size: 30px; font - weight: bold;}
    button { width: 145px; margin: 3px;}
  </style></head>
<body><p><button id = "myBtnTop">上端显示标题</button>
         <button id = "myBtnCenter">居中显示标题</button>
         <button id = "myBtnBottom">下端显示标题</button></p>
<div><span>经典桌面</span></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-box-pack：center 规定在水平方向上居中显示子元素。在 CSS3 中，box-pack 属性规定当框大于子元素的尺寸时在何处放置子元素，规定水平框中的水平位置以及垂直框中的垂直位置。box-pack 属性的语法格式如下。

```
box - pack: start | end | center | justify
```

其中，各属性值的说明如下。

（1）start：对于正常方向的框，首个子元素的左边缘被放在左侧（最后的子元素后是所有剩余的空间）；对于相反方向的框，最后子元素的右边缘被放在右侧（首个子元素前是所有剩余的空间）。

（2）end：对于正常方向的框，最后子元素的右边缘被放在右侧（首个子元素前是所有剩余的空间）；对于相反方向的框，首个子元素的左边缘被放在左侧（最后子元素后是所有剩余的空间）。

（3）center：该属性值均等地分割多余空间，其中一半空间被置于首个子元素前，另一半被置于最后一个子元素后。

（4）justify：该属性值在每个子元素之间分割多余的空间（首个子元素前和最后一个子元素后没有多余的空间）。

-webkit-box-align：center 规定在垂直方向上居中显示子元素。在 CSS3 中，box-align 属性规定如何对齐框的子元素，该属性的语法格式如下。

```
box - align: start | end | center | baseline | stretch
```

其中，各属性值的说明如下。

（1）start：对于正常方向的框，每个子元素的上边缘沿着框的顶边放置；对于反方向的框，每个子元素的下边缘沿着框的底边放置。

（2）end：对于正常方向的框，每个子元素的下边缘沿着框的底边放置；对于反方向的框，每个子元素的上边缘沿着框的顶边放置。

（3）center：该属性值均等地分割多余的空间，一半位于子元素之上，另一半位于子元素之下。

（4）baseline：该属性值在 box-orient 是 inline-axis 或 horizontal 时所有子元素均与其基线对齐。

（5）stretch：该属性值实现拉伸子元素以填充包含块。

此实例的源文件名是 myHtmlB193.html。

522　动态重置弹性盒中子元素的排列方式

此实例主要通过设置弹性盒子的 align-content 属性实现动态重置弹性盒中各子元素的排列方式。当在 Google Chrome 浏览器中显示该页面时，单击“顶部对齐”按钮，则弹性盒中各子元素的排列效果如图 522-1 所示；单击“中间对齐”按钮，则弹性盒中各子元素的排列效果如图 522-2 所示；单击

“底部对齐”按钮，则在底部对齐各子元素。有关此实例的主要代码如下。



图 522-1



图 522-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnTop").click(function() {              //顶部对齐
   $("#box").css("align-content", "flex-start");
  });
  $("#myBtnCenter").click(function() {       //中间对齐
   $("#box").css("align-content", "center");
  });
  $("#myBtnBottom").click(function() {       //底部对齐
   $("#box").css("align-content", "flex-end"); });});
 </script>
 <style>
  .box { display: -webkit-flex; -webkit-flex-wrap: wrap; width: 400px; height: 400px; margin: 0;
padding: 0; border-radius: 5px;list-style: none; background-color: #EEE;}
  .box li { margin: 5px; padding: 10px; text-align: center; }
  #box {/* -webkit-align-content:flex-end; */}
  img { width: 70px; height: 70px;}
 </style></head>
<body>
<p><input type="button" value="顶部对齐" id="myBtnTop" style="width:127px"/>
   <input type="button" value="中间对齐" id="myBtnCenter" style="width:127px"/>
 <input type="button" value="底部对齐" id="myBtnBottom" style="width:127px"/></p>
<ul id="box" class="box">
 <li><img src="img/B066A.png"/></li><li><img src="img/B066A.png"/></li>
 <li><img src="img/B066A.png"/></li><li><img src="img/B066A.png"/></li>
 <li><img src="img/B066A.png"/></li><li><img src="img/B066A.png"/></li>
</ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#box").css("align- content","flex-start")用于设置弹性盒中的所有子元素在顶部对齐。当弹性容器的侧轴还有多余空间时，此属性可以用来调整伸缩行在弹性容器里的对齐方式，此属性在只有一行的伸缩容器上没有效果。align-content 的语法格式如下。

```
align-content:flex-start | flex-end | center | space-between | space-around | stretch
```

其中，各属性值的意义如下。

(1) flex-start：表示各行向弹性盒容器的起始位置堆叠。在弹性盒容器中第一行的侧轴起始边界紧靠着该弹性盒容器的侧轴起始边界，之后的每一行都紧靠着前面一行。

(2) flex-end：表示各行向弹性盒容器的结束位置堆叠。在弹性盒容器中最后一行的侧轴起始边界紧靠着该弹性盒容器的侧轴结束边界，之后的每一行都紧靠着前面一行。

(3) center：表示各行向弹性盒容器的中间位置堆叠。各行两两紧靠住同时在弹性盒容器中居中对齐，保持弹性盒容器的侧轴起始内容边界和第一行之间的距离与该容器的侧轴结束内容边界和最后一行之间的距离相等。如果剩下的空间是负数，则各行会向两个方向溢出相等距离。

(4) space-between：表示各行在弹性盒容器中平均分布。如果剩余的空间是负数或在弹性盒容器中只有一行，该值等效于 flex-start。在其他情况下，第一行的侧轴起始边界紧靠着弹性盒容器的侧轴起始内容边界，最后一行的侧轴结束边界紧靠着弹性盒容器的侧轴结束内容边界，剩余的行则按一定方式在弹性盒窗口中排列，以保持两两之间的空间相等。

(5) space-around：表示各行在弹性盒容器中平均分布，两端保留子元素与子元素之间的间距大小的一半。如果剩余的空间是负数或在弹性盒容器中只有一行，该值等效于 center。在其他情况下，各行会按一定方式在弹性盒容器中排列，以保持两两之间的空间相等，同时第一行前面及最后一行后面的空间是其他空间的一半。

(6) stretch：表示各行将会伸展以占用剩余的空间。如果剩余的空间是负数，该值等效于 flex-start。在其他情况下，剩余空间被所有行平分，以扩大它们的侧轴尺寸。

此实例的源文件名是 myHtmlB080.html。

523　动态重置弹性盒中子元素的对齐方式

此实例主要通过设置弹性盒的 align-items 属性实现动态重置弹性盒中各子元素的对齐方式。当在 Google Chrome 浏览器中显示该页面时，单击“顶部对齐”按钮，则在弹性盒中各子元素的对齐效果如图 523-1 所示；单击“中间对齐”按钮，则在弹性盒中各子元素的对齐效果如图 523-2 所示；单击“底部对齐”按钮，则在弹性盒中各子元素的对齐效果如图 523-3 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnTop").click(function() {            //顶部对齐
     $("#box").css("align-items", "flex-start");
   });
   $("#myBtnCenter").click(function() {         //中间对齐
     $("#box").css("align-items", "center");
   });
   $("#myBtnBottom").click(function() {         //底部对齐
     $("#box").css("align-items", "flex-end"); });});
```

```
</script>
 <style>
  .box {display: -webkit-flex; -webkit-flex-wrap: wrap; width: 456px;height: 315px; margin: 0;
padding: 0; border-radius: 5px; list-style: none; background-color: #eee;}
  .box li { margin: 2px; padding: 2px; text-align: center;}
  #box { /* -webkit-align-items:flex-end; */}
  img { border-radius: 5px;}
  input { width: 147px;}
 </style></head>
<body><p><input type="button" value="顶部对齐" id="myBtnTop"/>
   <input type="button" value="中间对齐" id="myBtnCenter"/>
   <input type="button" value="底部对齐" id="myBtnBottom"/></p>
<ul id="box" class="box">
    <li><img src="img/B081A.jpg"/></li><li><img src="img/B081B.jpg"/></li>
    <li><img src="img/B081C.jpg"/></li></ul></body></html>
```

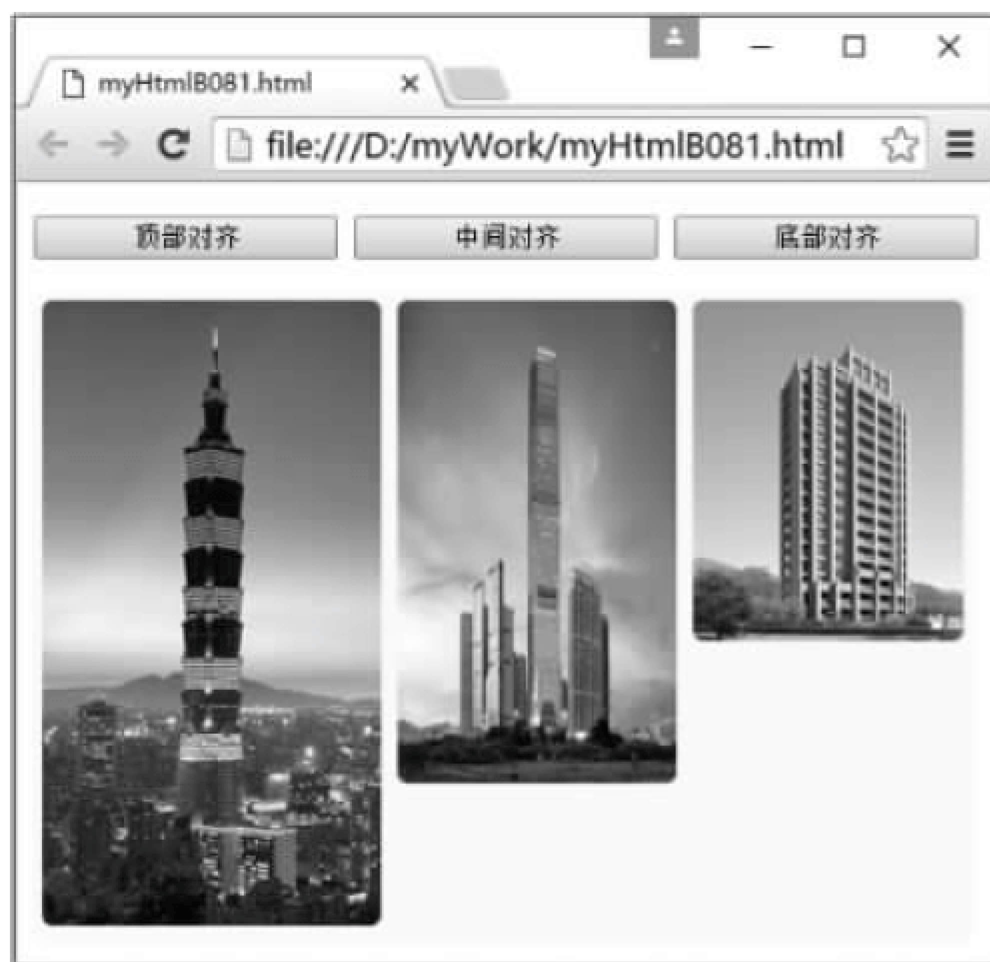


图 523-1



图 523-2

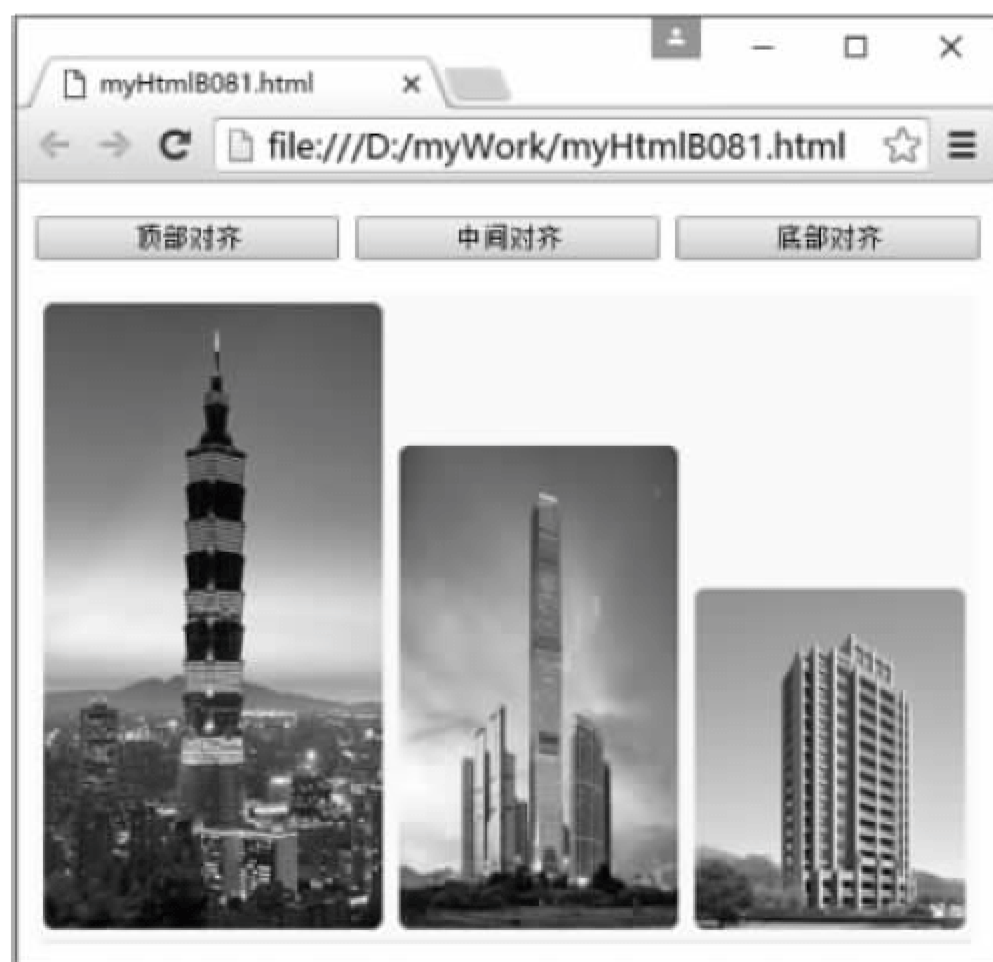


图 523-3

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#box").css("align-items","flex-start")用于设置弹性盒中的所有子元素在顶部对齐。在CSS3中，align-items属性用于定义弹性盒中的子元素在弹性盒的当前行的侧轴(纵轴)方向上的对齐方式，align-items的语法格式如下。

```
align-items: flex-start | flex-end | center | baseline | stretch
```

其中，各属性值的意义如下。

(1) flex-start：表示弹性盒中的子元素的侧轴(纵轴)起始位置的边界紧靠着该行的侧轴起始边界。

(2) flex-end：表示弹性盒中的子元素的侧轴(纵轴)起始位置的边界紧靠着该行的侧轴结束边界。

(3) center：表示弹性盒中的子元素在该行的侧轴(纵轴)上居中放置，如果该行的尺寸小于弹性盒中子元素的尺寸，则会向两个方向溢出相同的长度。

(4) baseline：表示如果弹性盒中的子元素的行内轴与侧轴为同一条，则该值与flex-start等效。在其他情况下，该值将参与基线对齐。

(5) stretch：表示如果指定侧轴大小的属性值为auto，则其值会使项目的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照min/max-width/height属性的限制。

此实例的源文件名是myHtmlB081.html。

524　设置弹性盒元素沿水平方向等距对齐

此实例主要通过设置弹性盒的justify-content属性实现动态重置弹性盒中的子元素沿水平方向的对齐方式。当在Google Chrome浏览器中显示该页面时，单击“向左对齐”按钮，则弹性盒中的各子元素的对齐效果如图524-1所示；单击“居中对齐”按钮，则弹性盒中的各子元素的对齐效果如图524-2所示；单击“向右对齐”按钮，则弹性盒中的各子元素的对齐效果如图524-3所示；单击“等距对齐”按钮，则弹性盒中的各子元素的对齐效果如图524-4所示。有关此实例的主要代码如下。



图　524-1



图　524-2



图　524-3



图　524-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnLeft").click(function() {           //向左对齐
    $("#box").css("justify-content", "flex-start");
   });
   $("#myBtnCenter").click(function() {         //居中对齐
    $("#box").css("justify-content", "center");
   });
   $("#myBtnRight").click(function() {          //向右对齐
    $("#box").css("justify-content", "flex-end");
   });
   $("#myBtnBetween").click(function() {        //等距对齐
    $("#box").css("justify-content", "space-between"); });});
 </script>
 <style>
  .box {display: -webkit-flex; -webkit-flex-wrap: wrap; width: 456px;
        height: 60px; margin: 0; padding: 0; border-radius: 5px;
        list-style: none; background-color: lightblue;}
  .box li { margin: 2px; padding: 2px; text-align: center;}
  img { width: 50px; height:50px;}
  input { width: 108px;}
 </style></head>
<body><p><input type="button" value="向左对齐" id="myBtnLeft"/>
    <input type="button" value="居中对齐" id="myBtnCenter"/>
    <input type="button" value="向右对齐" id="myBtnRight"/>
    <input type="button" value="等距对齐" id="myBtnBetween"/></p>
<ul id="box" class="box">
    <li><img src="img/B082.png"/></li><li><img src="img/B082.png"/></li>
    <li><img src="img/B082.png"/></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#box").css("justify-content","flex-start")用于设置弹性盒中的所有子元素靠左对齐。在CSS3中，justify-content属性用于设置或检索弹性盒中的子元素在主轴(横轴)方向上的对齐方式，当弹性盒中一行上的所有子元素都不能伸缩或已经达到其最大值时，这一属性可协助对多余的空间进行分配，当元素溢出某行时这一属性同样会在对齐上进行控制。justify-content的语法格式如下。

```
justify-content: flex-start | flex-end | center | space-between | space-around
```

其中，各属性值的意义如下。

(1) flex-start：表示弹性盒中的子元素将向行起始位置对齐。该行的第一个子元素的主起始位置的边界将与该行的主起始位置的边界对齐，同时所有后续的子元素与其前一个子元素对齐。

(2) flex-end：表示弹性盒中的子元素将向行结束位置对齐。该行的第一个子元素的主结束位置的边界将与该行的主结束位置的边界对齐，同时所有后续的子元素与其前一个子元素对齐。

(3) center：表示弹性盒中的子元素将向行中间位置对齐。该行的子元素将相互对齐并在行中居中对齐，同时第一个元素与行的主起始位置的边距等于最后一个元素与行的主结束位置的边距，如果剩余空间是负数，则保持两端相等长度的溢出。

(4) space-between：表示弹性盒中的子元素会平均地分布在行里。如果最左边的剩余空间是负数，或该行只有一个子元素，则该值等效于flex-start。在其他情况下，第一个元素的边界与行的主起始位置的边界对齐，同时最后一个元素的边界与行的主结束位置的边距对齐，而剩余的子元素平均分

布，并确保两两之间的空白空间相等。

(5) space-around：表示弹性盒中的子元素会平均地分布在行里，两端保留子元素与子元素之间的间距大小的一半。如果最左边的剩余空间是负数，或该行只有一个子元素，则该值等效于 center。在其他情况下，子元素平均分布，并确保两两之间的空白空间相等，同时第一个元素前的空间以及最后一个元素后的空间为其他空白空间的一半。

此实例的源文件名是 myHtmlB082.html。

525 设置弹性盒元素沿水平或垂直方向布局

此实例主要通过设置弹性盒的 box-orient 属性实现动态重置弹性盒中的子元素沿水平或垂直方向布局。当在 Google Chrome 浏览器中显示该页面时，单击“水平布局”按钮，则弹性盒中的各子元素的布局效果如图 525-1 所示；单击“垂直布局”按钮，则弹性盒中的各子元素的布局效果如图 525-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnHorizontal").click(function() {      //水平布局
    $("#box").css("box-orient", "horizontal");
   });
   $("#myBtnVertical").click(function() {        //垂直布局
    $("#box").css("box-orient", "vertical"); });});
 </script>
 <style type="text/css">
  .box {display: -webkit-box; width: 600px; height: 380px;
        margin: 0;padding: 0;list-style: none;}
  #box {/* -webkit-box-orient: vertical; */}
  .box li:nth-child(1) { -webkit-box-flex: 5; background: lightblue;}
  .box li:nth-child(2) { -webkit-box-flex: 5; background: lightpink;}
  .box li:nth-child(3) { -webkit-box-flex: 4; background: lightgreen;}
  li { padding: 10px; }
  h3 { text-align: center}
  input {width: 295px;}
 </style></head>
<body><p><input type="button" value="水平布局" id="myBtnHorizontal"/>
   <input type="button" value="垂直布局" id="myBtnVertical"/></p>
<ul id="box" class="box"><li><h3>红藕香残玉簟秋</h3>
     <p>红藕香残玉簟秋。轻解罗裳，独上兰舟。云中谁寄锦书来，雁字回时，月满西楼。花自飘零水自流。一种相思，两处闲愁。此情无计可消除，才下眉头，却上心头。</p></li>
 <li><h3>春晚</h3>
     <p>风住尘香花已尽，日晚倦梳头。物是人非事事休，欲语泪先流。闻说双溪春尚好，也拟泛轻舟。只恐双溪舴艋舟，载不动许多愁。</p></li>
 <li><h3>寻寻觅觅</h3>
     <p>寻寻觅觅，冷冷清清，凄凄惨惨戚戚。乍暖还寒时候，最难将息。三杯两盏淡酒，怎敌他、晚来风急?雁过也，正伤心，却是旧时相识。满地黄花堆积。憔悴损，如今有谁堪摘?守着窗儿，独自怎生得黑?梧桐更兼细雨，到黄昏、点点滴滴。这次第，怎一个愁字了得!</p></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("#box").css("box-orient", "horizontal")用于设置弹性盒中的所有子元素按照水平方向进行布局。在 CSS3 中，box-orient 属性用于设置或检索弹性盒中的子元素的排列方式。box-orient 属性的语法格式如下。

```
box-orient:horizontal | vertical | inline-axis | block-axis
```

其中，各属性值的意义如下。

（1）horizontal：设置弹性盒对象的子元素从左到右水平排列。

（2）vertical：设置弹性盒对象的子元素从上到下纵向排列。

（3）inline-axis：设置弹性盒对象的子元素沿行轴排列。

（4）block-axis：设置弹性盒对象的子元素沿块轴排列。

此实例的源文件名是 myHtmlB083.html。



图 525-1

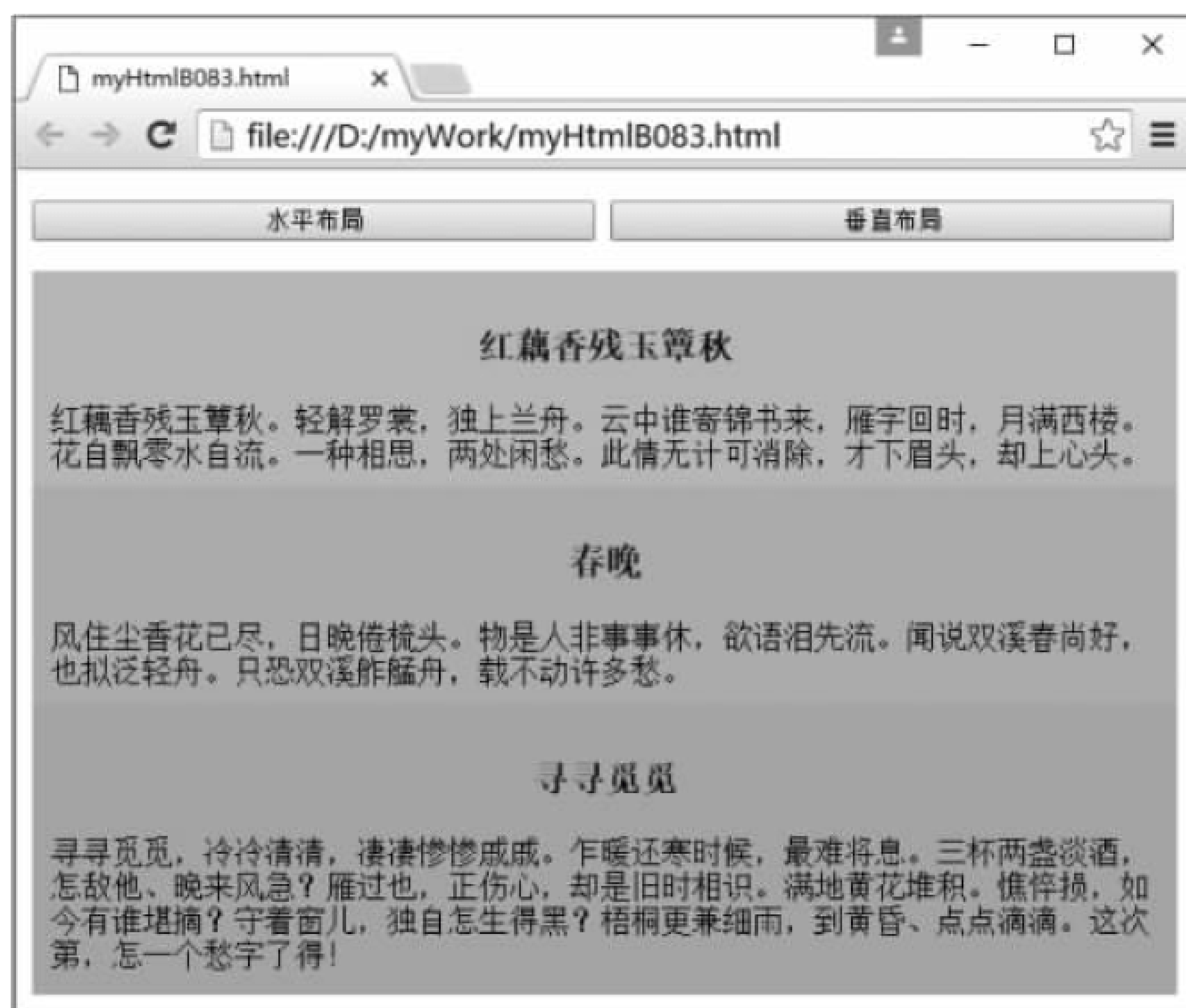


图 525-2

526 根据收缩因子分配弹性盒的子元素空间

此实例主要通过设置弹性盒子元素的 flex-shrink 属性实现根据收缩因子分配弹性盒的子元素空间。当在 Google Chrome 浏览器中显示该页面时，弹性盒中的 3 幅图片(3 个子元素)将根据收缩因子分配空间，如图 526-1 所示。有关此实例的主要代码如下。

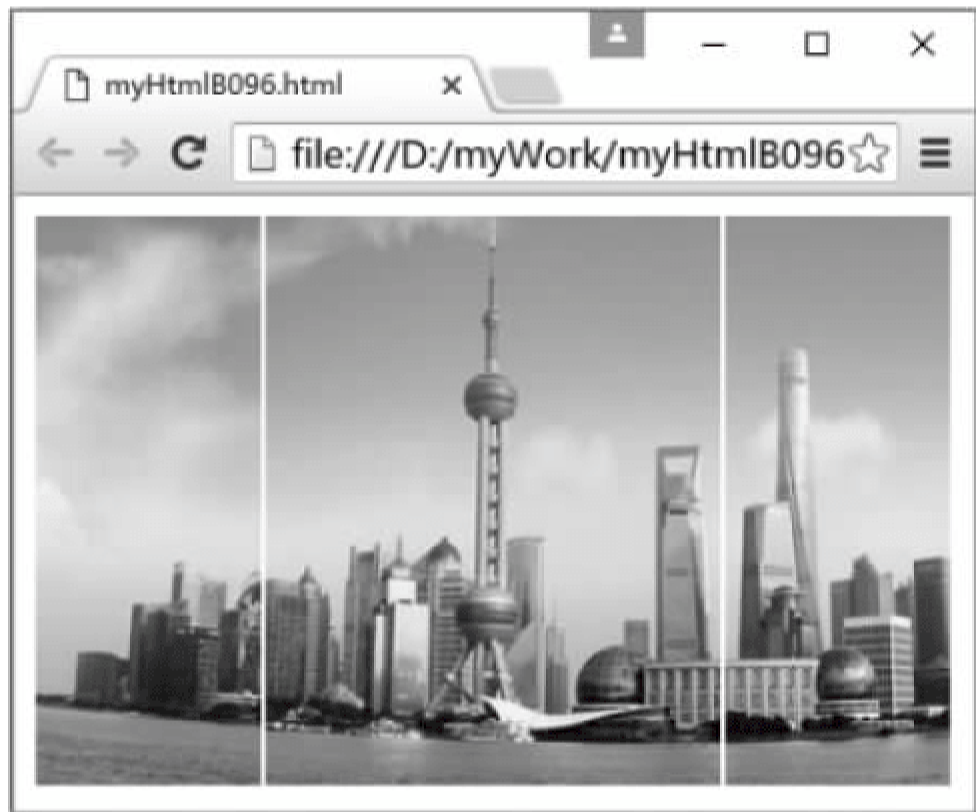


图 526-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   #myContainer { display: -webkit-flex;width: 400px;height:250px;
                  margin: 0;padding: 0;list-style: none;}
   #myContainer li {width: 250px;  margin:1px;}
   #myContainer li:nth-child(1) { -webkit-flex-shrink: 2;
                                 background: gray;background-image: url(img/B096A.jpg);}
   #myContainer li:nth-child(2) {background: lightpink;
                                background-image: url(img/B096B.jpg);}
   #myContainer li:nth-child(3) { -webkit-flex-shrink: 2;
                                 background: lightblue; background-image: url(img/B096C.jpg);}
 </style></head>
<body><ul id="myContainer"><li></li><li></li><li></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，flex-shrink 属性用于设置或检索弹性盒的收缩比率，弹性盒将根据收缩比率来分配(收缩)子元素的空间。flex-shrink 的默认值为 1，如果没有显示定义该属性，将会自动按照默认值 1 在所有因子相加之后计算比率来进行空间收缩。例如，弹性盒中有 a、b、c 几个子元素，c 显式定义了 flex-shrink 属性值为 3，a、b 没有显式定义，因此将根据默认值 1 来计算，可以看到总共将剩余空间分成了 5 份，其中 a 占 1 份、b 占 1 份、c 占 3 份，即 1∶1∶3。如果父容器(弹性盒)定义为 400px，子项(子元素)被定义为 200px，相加之后即为 600px，超出父容器 200px。那么超出的 200px 需要被 a、b、c 消化，根据收缩因子加权综合可得 200×1+200×1+200×3=1000px。于是可以计算 a、b、c 将被移除的溢出量。

a 被移除溢出量：(200×1/1000)×200，即等于 40px；

b 被移除溢出量：(200×1/1000)×200，即等于 40px；

c 被移除溢出量：(200×3/1000)×200，即等于 120px。

最后 a、b、c 的实际宽度分别为 200－40＝160px、200－40＝160px、200－120＝80px。

此实例的源文件名是 myHtmlB096.html。

527 根据扩展因子分配弹性盒的子元素空间

此实例主要通过设置弹性盒子元素的 flex-grow 属性实现根据扩展因子分配弹性盒的子元素空间。当在 Google Chrome 浏览器中显示该页面时，在弹性盒中的 3 幅图片(3 个子元素)将根据扩展因子分配空间，如图 527-1 所示。有关此实例的主要代码如下。

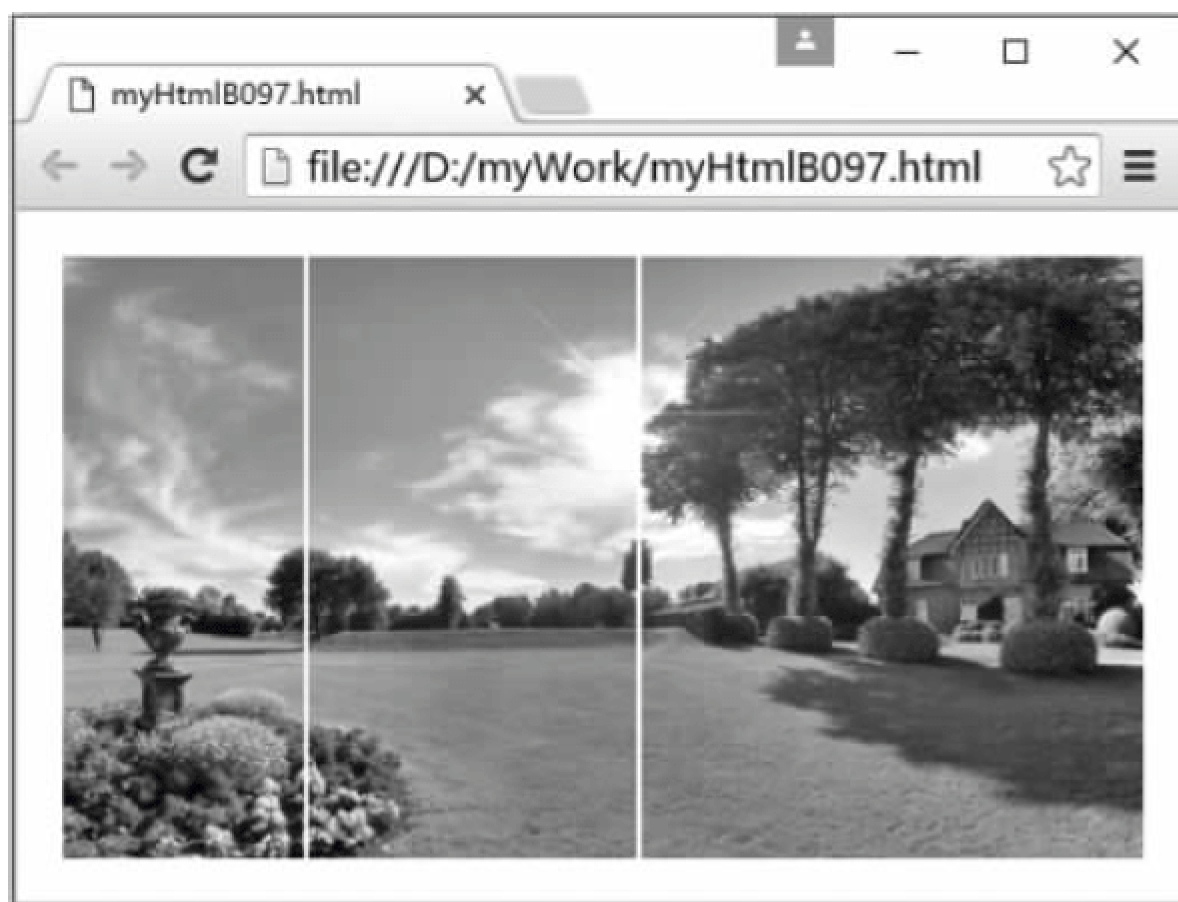


图 527-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myBox {display: -webkit-flex; height: 250px;width: 450px;
         margin: 0; padding: 10px;list-style: none;}
  #myBox li {width: 100px;height: 250px;margin: 1px;}
  #myBox li:nth-child(1) {background-image: url(img/B097A.jpg);}
  #myBox li:nth-child(2) { -webkit-flex-grow: 1;
                          background-image: url(img/B097B.jpg);}
  #myBox li:nth-child(3) { -webkit-flex-grow: 3;
                          background-image: url(img/B097C.jpg);}
 </style></head>
<body><ul id="myBox"><li></li><li></li><li></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，flex-grow 属性用于设置或检索弹性盒的扩展比率，并根据弹性盒子元素所设置的扩展因子作为比率来分配剩余空间。flex-grow 属性的默认值为 0，如果没有显示定义该属性，是不会拥有分配剩余空间权利的。在此实例中，第二个和第三个子元素都显式定义了 flex-grow 属性值，即弹性盒的剩余空间被分成了 4 份，其中第二个元素占 1 份，第三个元素占 3 份，即 1∶3。

此实例的源文件名是 myHtmlB097.html。

528 使用弹性盒控制文本始终显示在正中间

此实例主要通过设置弹性盒的 justify-content 属性和 align-items 属性实现控制文本始终显示在容器中间。当在 Google Chrome 浏览器中显示该页面时，文本内容始终显示在图片的正中间，如图 528-1 所示。有关此实例的主要代码如下。



图 528-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  .box{ background: #0099CC; width: 400px; height: 250px; background - image: url(img/B098.jpg);
        background - repeat: no - repeat; background - size: 100% 100%; display: flex; justify -
        content: center; align - items:center;}
  p{text - indent: 35px; width:350px;color:yellow; }
 </style></head>
<body><div class = "box"><p>重庆中央公园,位于西江新区同茂大道和节庆大道之间,是目前国内最大的开
放式城市中心公园。融会中西方文化,依托重庆山水风貌特色,体现自然和谐之美的现代城市公园,主要包含了
中央广场、活力水景、阳光草坡、半岛镜湖和密林溪流五大景区。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：flex 表示将对象作为弹性伸缩盒显示；justify-content：center 表示弹性盒的子元素将向行中间位置对齐；align-items：center 表示弹性盒的子元素在该行的侧轴（纵轴）上居中放置。

此实例的源文件名是 myHtmlB098.html。

529 指定子元素分配弹性盒的纵向剩余空间

此实例主要通过设置元素的 display、flex-direction、flex 等属性实现指定子元素分配弹性盒的纵向剩余空间。当在 Google Chrome 浏览器中显示该页面时，单击"首项分配纵向剩余空间"按钮，则第一项（第一幅图片"武隆风光"）分配弹性盒的纵向全部剩余空间，第二项和第三项以最小高度值（20px）显示，如图 529-1 所示；单击"两项均分纵向剩余空间"按钮，则第一项、第二项（第一幅图片"武隆风光"、第二幅图片"大理风光"）平均分配弹性盒的纵向剩余空间，第三项以最小高度值（20px）显示，如图 529-2 所示；单击"三项均分纵向剩余空间"按钮，则 3 幅图片平均分配弹性盒的纵向剩余空

间，如图 529-3 所示。有关此实例的主要代码如下。



图 529-1



图 529-2



图 529-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnOne").click(function() {            //首项分配纵向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "0");
    $("#myImagec").css("flex", "0");
   });
   $("#myBtnTwo").click(function() {            //两项均分纵向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "1");
    $("#myImagec").css("flex", "0");
   });
   $("#myBtnThree").click(function() {          //三项均分纵向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "1");
    $("#myImagec").css("flex", "1"); });});
```

```
    </script>
    <style type="text/css">
      #container{display: flex;flex-direction:column; border: solid 1px snow;width: 450px; height:
                 250px; background-color: lightgoldenrodyellow; }
      #myImagea{ background-image: url(img/B099A.jpg); }
      #myImageb{ background-image: url(img/B099B.jpg);}
      #myImagec{ background-image: url(img/B099C.jpg);}
      #myImagea, #myImageb, #myImagec{font-family: 楷体; font-size: 20px;
                                   padding: 5px;  color:yellow;margin:1px;  min-height:20px; }
      input{width:144px;}
    </style></head>
   <body><input type="button" value="首项分配纵向剩余空间" id="myBtnOne"/>
   <input type="button" value="两项均分纵向剩余空间" id="myBtnTwo"/>
   <input type="button" value="三项均分纵向剩余空间" id="myBtnThree"/>
   <div id="container">
    <div id="myImagea">武隆风光</div>
    <div id="myImageb">大理风光</div>
    <div id="myImagec">长寿风光</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：flex 表示将对象作为弹性伸缩盒显示；flex-direction:column 表示弹性盒的子元素按照纵向从上往下排列；flex 属性用于设置或检索弹性盒模型对象的子元素如何分配空间，在 JavaScript 中的语法格式是 object.style.flex＝"1"，即具体数值应该加引号。

此实例的源文件名是 myHtmlB099.html。

530　指定子元素分配弹性盒的横向剩余空间

此实例主要通过设置元素的 display、flex-direction、flex 等属性实现指定子元素分配弹性盒的横向剩余空间。当在 Google Chrome 浏览器中显示该页面时，单击"首项分配横向剩余空间"按钮，则第一项（第一幅图片"武隆风光"）分配弹性盒的横向（水平方向）全部剩余空间，第二项和第三项以最小宽度值（20px）显示，如图 530-1 所示；单击"两项均分横向剩余空间"按钮，则第一项、第二项（第一幅图片"武隆风光"、第二幅图片"大理风光"）平均分配弹性盒的横向剩余空间，第三项以最小宽度值（20px）显示，如图 530-2 所示；单击"三项均分横向剩余空间"按钮，则 3 幅图片平均分配弹性盒的横向剩余空间，如图 530-3 所示。有关此实例的主要代码如下。



图　530-1



图 530-2



图 530-3

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnOne").click(function() {        //首项分配横向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "0");
    $("#myImagec").css("flex", "0");
   });
   $("#myBtnTwo").click(function() {        //两项均分横向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "1");
    $("#myImagec").css("flex", "0");
   });
   $("#myBtnThree").click(function() {      //三项均分横向剩余空间
    $("#myImagea").css("flex", "1");
    $("#myImageb").css("flex", "1");
    $("#myImagec").css("flex", "1"); });});
 </script>
 <style type="text/css">
  #container{display: flex; flex-direction:row; border: solid 1px snow; width: 450px; height: 250px;
           background-color: lightgoldenrodyellow;}
  #myImagea{ background-image: url(img/B099A.jpg);}
  #myImageb{background-image: url(img/B099B.jpg);}
  #myImagec{background-image: url(img/B099C.jpg);}
  #myImagea, #myImageb, #myImagec{ font-family: 楷体;font-size: 20px;
                              padding: 5px; color:yellow; margin:1px; min-width:50px;}
  input{width:144px;}
 </style></head>
<body><input type="button" value="首项分配横向剩余空间" id="myBtnOne"/>
<input type="button" value="两项均分横向剩余空间" id="myBtnTwo"/>
<input type="button" value="三项均分横向剩余空间" id="myBtnThree"/>
<div id="container"><div id="myImagea">武隆风光</div>
 <div id="myImageb">大理风光</div>
 <div id="myImagec">长寿风光</div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：flex 表示将对象作为弹性伸缩盒显示；flex-direction：row 表示弹性盒的子元素按照横向从左往右排列，flex 属性用于设置或检索

弹性盒模型对象的子元素如何分配空间，在 JavaScript 中的语法格式是 object. style. flex="1"，即具体数值应该加引号。注意，如果元素不是弹性盒模型对象的子元素，则 flex 属性不起作用。

此实例的源文件名是 myHtmlB100. html。

531　允许弹性盒的子元素具有自动换行功能

此实例主要通过设置弹性盒元素的 display、flex-wrap 等属性实现允许弹性盒的子元素具有自动换行功能。当在 Google Chrome 浏览器中显示该页面时，拖动浏览器的边框线改变页面宽度，则页面（弹性盒）中的小图片（子元素）在一行不能显示时将自动换到下一行显示，如图 531-1 和图 531-2 所示。有关此实例的主要代码如下。



图　531-1

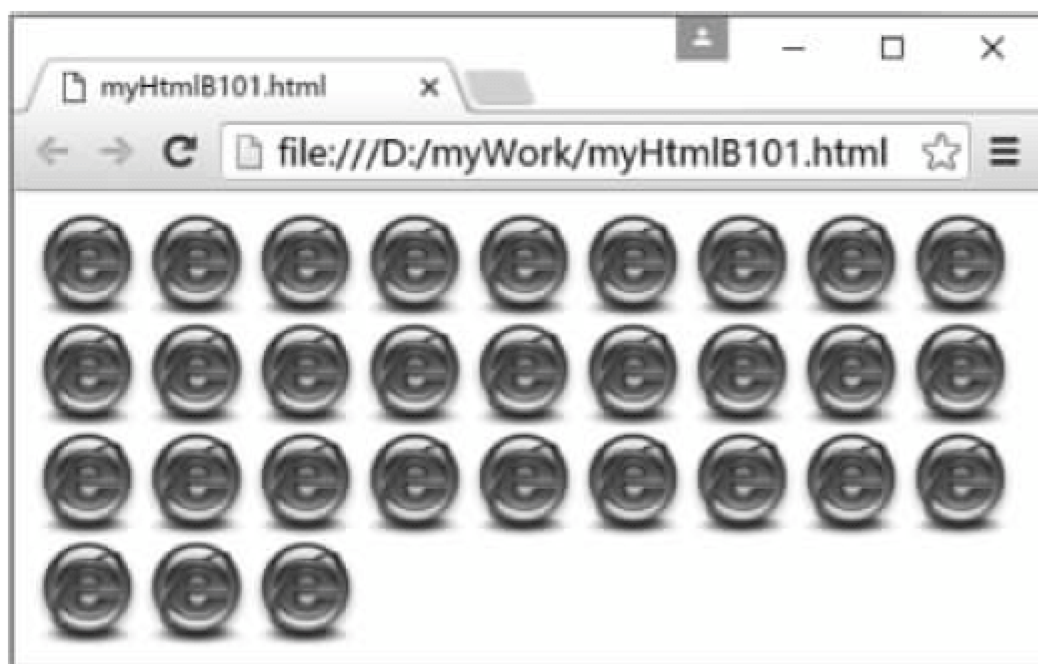


图　531-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {//添加 30 幅图片
   var myImage="<img src='img/B101.png' width='50px' height='50px' />";
   for(i=0;i<30;i++){   $("body").append(myImage); } });
 </script>
 <style type="text/css">
  body{ display:-webkit-flex; -webkit-flex-wrap:wrap;}
 </style></head>
<body></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display：flex 表示将对象作为弹性盒显示；flex-wrap 属性用于控制弹性盒的子元素是单行还是多行显示，同时横轴的方向决定了新行堆叠的方向。flex-wrap 属性的语法格式如下。

```
flex-wrap:nowrap | wrap | wrap-reverse
```

其中，nowrap 表示弹性盒的子元素只在一行中显示，在该情况下子元素可能会溢出弹性盒；wrap 表示可以多行显示弹性盒的子元素，在此情况下弹性盒子元素的溢出部分会被放置到新行，子项内部会发生断行，即换行；wrap-reverse 表示反转 wrap 排列，即如果 wrap 排列是从开始到结束，则 wrap-reverse 将从结束到开始。

此实例的源文件名是 myHtmlB101. html。

532 纵向拉伸对齐弹性盒中的各个子元素

此实例主要通过设置弹性盒子元素的 align-self 属性实现纵向拉伸对齐弹性盒中的各个子元素。当在 Google Chrome 浏览器中显示该页面时，单击“居中对齐”按钮，则弹性盒中的各个子元素将位于弹性盒的中间，如图 532-1 所示；单击“拉伸对齐”按钮，则在弹性盒中的各个子元素将纵向拉伸，纵向对齐弹性盒，如图 532-2 所示。有关此实例的主要代码如下。



图 532-1



图 532-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnCenter").click(function() {          //居中对齐
     $("li").css("align-self", " center");
  });
   $("#myBtnStretch").click(function() {         //拉伸对齐
     $("li").css("align-self", "stretch"); });});
 </script>
 <style>
  .box { display: -webkit-flex; -webkit-align-items: flex-end;
         height: 250px; margin: 0; padding: 10px; border-radius: 5px;
         list-style: none;background-color: #EEE;}
  .box li { margin: 5px; padding: 10px; border-radius: 5px; background: #AAA; text-align: center;
background-repeat: no-repeat; background-size: 100% 100%;width: 100px; min-height: 100px;}
  .box li:nth-child(1){ background-image: url(img/B102A.jpg); }
  .box li:nth-child(2){ background-image: url(img/B102B.jpg); }
  .box li:nth-child(3){ background-image: url(img/B102C.jpg); }
  input { width: 193px; }
 </style></head>
<body><p><input type="button" value="居中对齐" id="myBtnCenter"/>
   <input type="button" value="拉伸对齐" id="myBtnStretch"/></p>
<ul id="box" class="box"><li></li><li></li><li></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，$("li").css("align-self",

"stretch")用于设置弹性盒中的所有子元素在纵向拉伸对齐。在 CSS3 中，align-self 用于定义弹性盒的子元素单独在侧轴(纵轴)方向上的对齐方式。align-self 的语法格式如下。

```
align-self:auto | flex-start | flex-end | center | baseline | stretch
```

其中，各属性值的意义如下。

(1) flex-start：表示弹性盒元素的侧轴(纵轴)起始位置的边界紧靠着该行的侧轴起始边界。

(2) flex-end：表示弹性盒元素的侧轴(纵轴)起始位置的边界紧靠着该行的侧轴结束边界。

(3) center：表示弹性盒元素在该行的侧轴(纵轴)上居中放置(如果该行的尺寸小于弹性盒元素的尺寸，则会向两个方向溢出相同的长度)。

(4) baseline：表示如果弹性盒元素的行内轴与侧轴为同一条，则该值与 flex-start 等效。在其他情况下，该值将与基线对齐。

(5) stretch：表示如果指定侧轴大小的属性值为 auto，则其值会使元素的边距盒的尺寸尽可能接近所在行的尺寸，但同时会遵照 min/max-width/height 属性的限制。

(6) auto：表示如果 align-self 的值为 auto，则其计算值为元素的父元素的 align-items 值，如果其没有父元素，则计算值为 stretch。

此实例的源文件名是 myHtmlB102.html。

533　根据比例分配弹性盒的子元素剩余空间

此实例主要通过设置弹性盒子元素的 flex-basis 属性实现子元素根据比例分配弹性盒的剩余空间。当在 Google Chrome 浏览器中显示该页面时，弹性盒中的两个子元素(图片)将按照 1∶2 的比例分配弹性盒的剩余空间，如图 533-1 所示。有关此实例的主要代码如下。

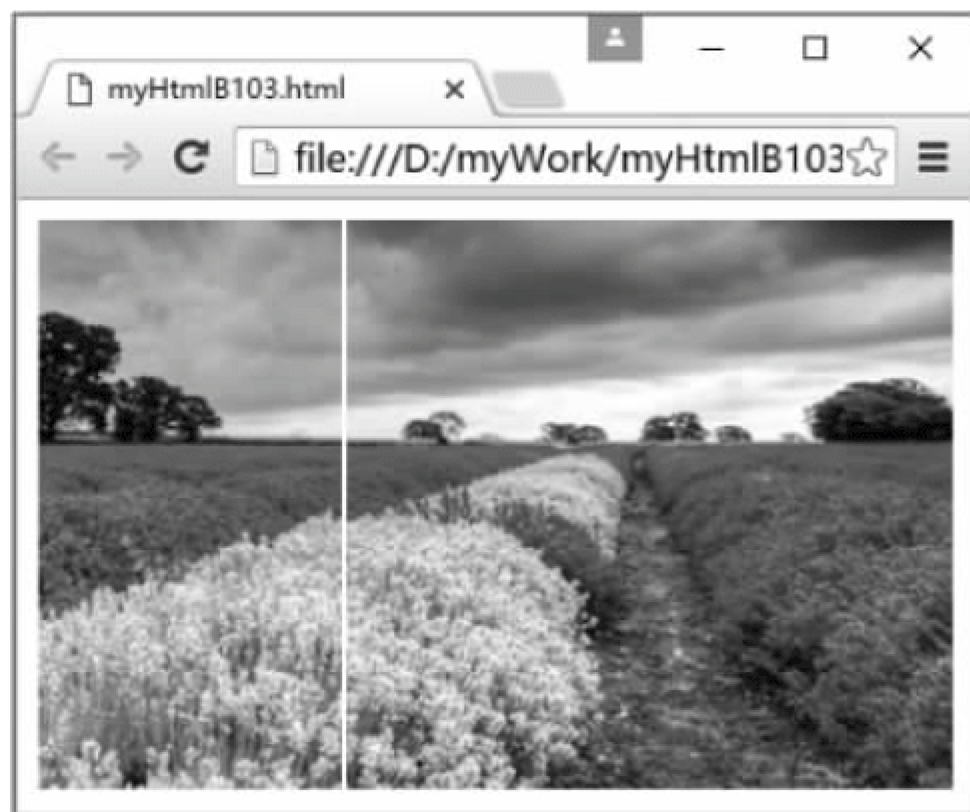


图　533-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myContainer {display:flex;width: 400px;height: 250px;
                margin: 0;padding: 0;list-style: none;}
  #myContainer li:nth-child(1) {flex-basis: 200px;
                                background-image: url(img/B103A.jpg);}
  #myContainer li:nth-child(2) {flex-basis: 400px;
                                background-image: url(img/B103B.jpg);}
```

```
    #myContainer li {margin: 1px; background-repeat: no-repeat;
                     background-size: 100% 100%;}
  </style></head>
<body><ul id="myContainer"><li></li><li></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，flex-basis 属性用于设置或检索弹性盒的伸缩基准值，如果所有子元素的基准值之和大于剩余空间，则会根据每个子元素设置的基准值按比例伸缩剩余空间。在此实例中，第一个子元素（图片）的宽度是 267px，第二个子元素（图片）的宽度是 133px，因此按照如下规则分配弹性盒的宽度。

第一个子元素（图片）的宽度＝400×(400/(400＋200))＝266.666666667；

第二个子元素（图片）的宽度＝400×(200/(400＋200))＝133.333333333。

flex-basis 属性的语法格式如下。

```
flex-basis:<length> | <percentage> | auto | content
```

其中，属性值<length>表示用长度值来定义宽度，不允许为负值；属性值<percentage>表示用百分比来定义宽度，不允许为负值；属性值 auto 表示无特定宽度值，取决于其他属性值；属性值 content 表示基于内容自动计算宽度。

此实例的源文件名是 myHtmlB103.html。

534　保持子元素在水平方向上始终位于容器的正中

此实例主要设置容器元素的 width 属性为-webkit-fit-content，从而实现保持子元素在水平方向上始终位于容器的正中。当在 Google Chrome 浏览器中显示该页面时，无论怎样拖动浏览器的右边框线改变窗口大小，图片总是随之改变，并且在水平方向上始终位于正中间，如图 534-1 所示。有关此实例的主要代码如下。

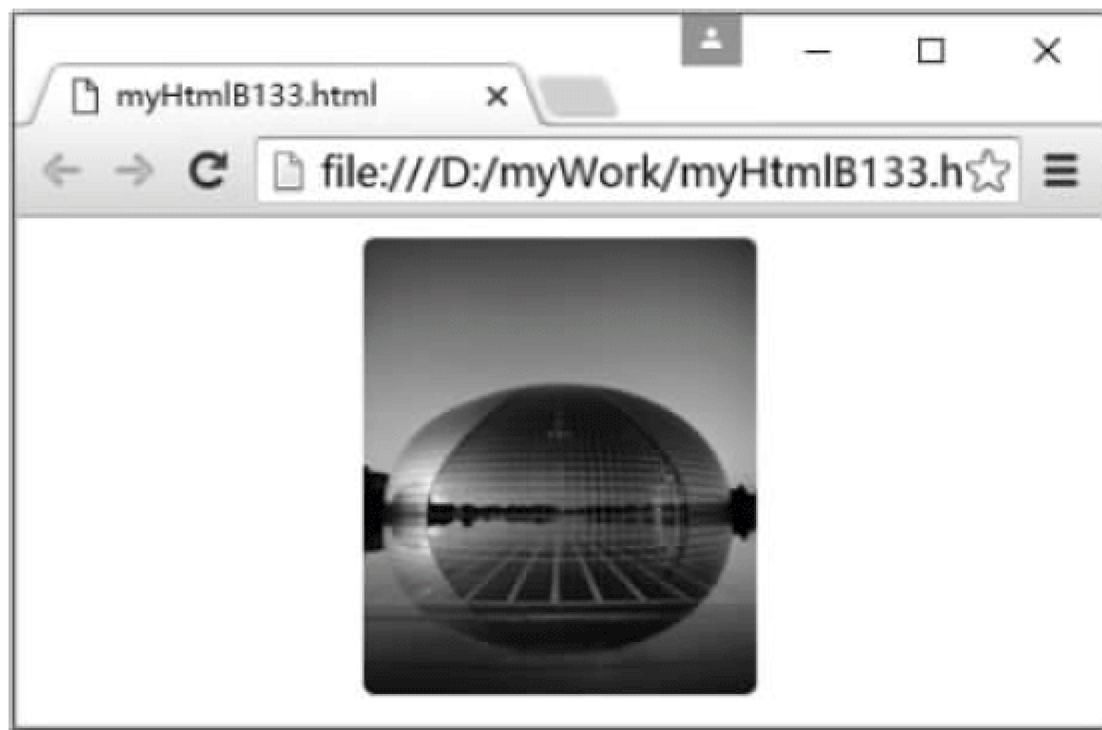


图　534-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .myContainer { margin: auto;  width: -webkit-fit-content; }
  img{ border-radius: 5px;}
 </style></head>
<body><div class="myContainer"><img src="img/B005B.jpg" ></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，-webkit-fit-content 是元素的 width 属性的关键字，它可以在实现元素收缩效果的同时保持原本的 block 水平状态，于是就可以直接使用 margin:auto 实现元素向内自适应居中效果。

此实例的源文件名是 myHtmlB133.html。

535 通过 object-position 属性控制子元素在容器中的位置

此实例主要通过设置子元素的 object-position 属性实现根据 left、right、top、bottom 控制子元素相对容器四边的位置。当在 Google Chrome 浏览器中显示该页面时，单击"右下角定位"按钮，则图像将显示在距离容器下边和右边 10px 的位置，如图 535-1 所示；单击"右上角定位"按钮，则图像将显示在距离容器上边和右边 10px 的位置，如图 535-2 所示；单击其他按钮，图像则会按照按钮标题所示的定位进行显示。有关此实例的主要代码如下。



图 535-1



图 535-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnLeftTop").click(function() {          //左上角定位
    $("img").css("object-position", "left 10px top 10px");
   });
   $("#myBtnLeftBottom").click(function() {       //左下角定位
    $("img").css("object-position", "left 10px bottom 10px");
   });
   $("#myBtnRightTop").click(function() {         //右上角定位
    $("img").css("object-position", "right 10px top 10px");
   });
   $("#myBtnRightBottom").click(function() {      //右下角定位
    $("img").css("object-position", "right 10px bottom 10px"); });});
 </script>
 <style>
  div { width: 425px; height: 250px; background-color: darkgreen;
       border-radius: 5px; }
  img { width: 100%; height: 100%;border-radius: 5px;object-fit: none;
       object-position: 50% 50% }
```

```
  input { width: 100px;}
 </style></head>
<body><p><input type="button" value="左上角定位" id="myBtnLeftTop"/>
   <input type="button" value="左下角定位" id="myBtnLeftBottom"/>
   <input type="button" value="右上角定位" id="myBtnRightTop"/>
   <input type="button" value="右下角定位" id="myBtnRightBottom"/></p>
<div><img src="img/B005B.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，object-position：right 10px bottom 10px 表示在距离容器右边和下边 10px 的位置显示子元素，object-position 的默认值是 50% 50%，也就是居中显示子元素。

此实例的源文件名是 myHtmlB138.html。

536 在水平方向上居中显示容器中的子元素

此实例主要设置容器子元素的 display 属性为 block、margin 属性为 auto，从而实现在水平方向上居中显示容器中的子元素。当在 Google Chrome 浏览器中显示该页面时，无论怎样拖动浏览器的右边框线改变窗口大小，图片的位置总是随之改变，并且在水平方向上始终位于正中间，如图 536-1 所示。有关此实例的主要代码如下。

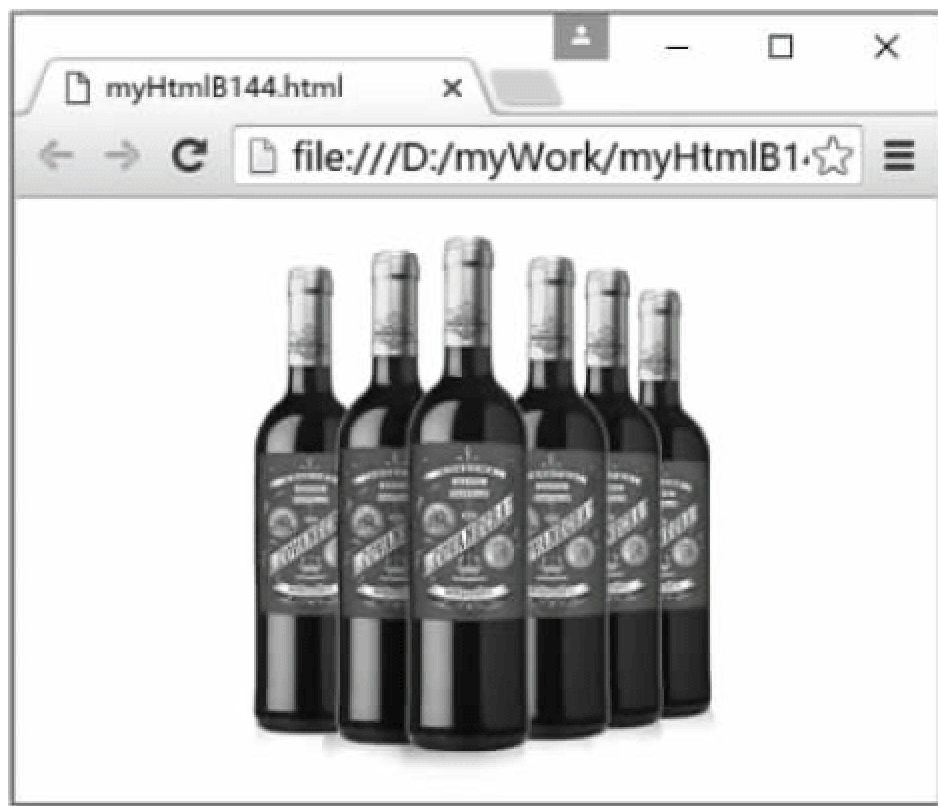


图 536-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  img{display: block;margin:auto; }
 </style></head>
<body><div><img src="img/B105.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，display 属性规定元素应该生成的盒子类型，display：block 表示此元素将显示为块级元素，此元素前后可以带有换行符
。margin 简写属性用于在一个声明中设置所有外边距属性，该属性可以有 1 到 4 个值，但是如果元素的 display 为 block，则设置元素的 margin 为 auto，将会使元素在水平方向上始终居中显示。

此实例的源文件名是 myHtmlB144.html。

537 自定义子元素在父容器中的自适应模式

此实例主要通过设置子元素的 object-fit 属性实现定制子元素(图像)在父容器中的自适应模式。当在 Google Chrome 浏览器中显示该页面时,单击"Contain 模式"按钮,则图像的显示效果如图 537-1 所示;单击"None 模式"按钮,则图像的显示效果如图 537-2 所示;单击其他按钮,图像则会按照按钮标题所示的模式进行显示。有关此实例的主要代码如下。

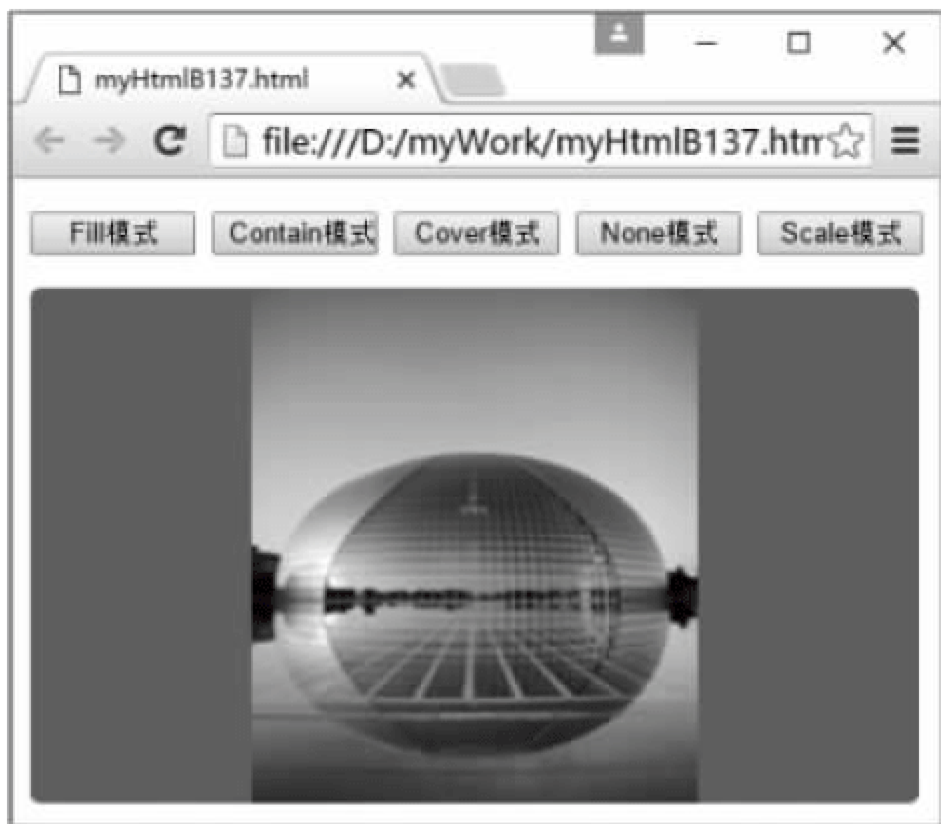


图 537-1

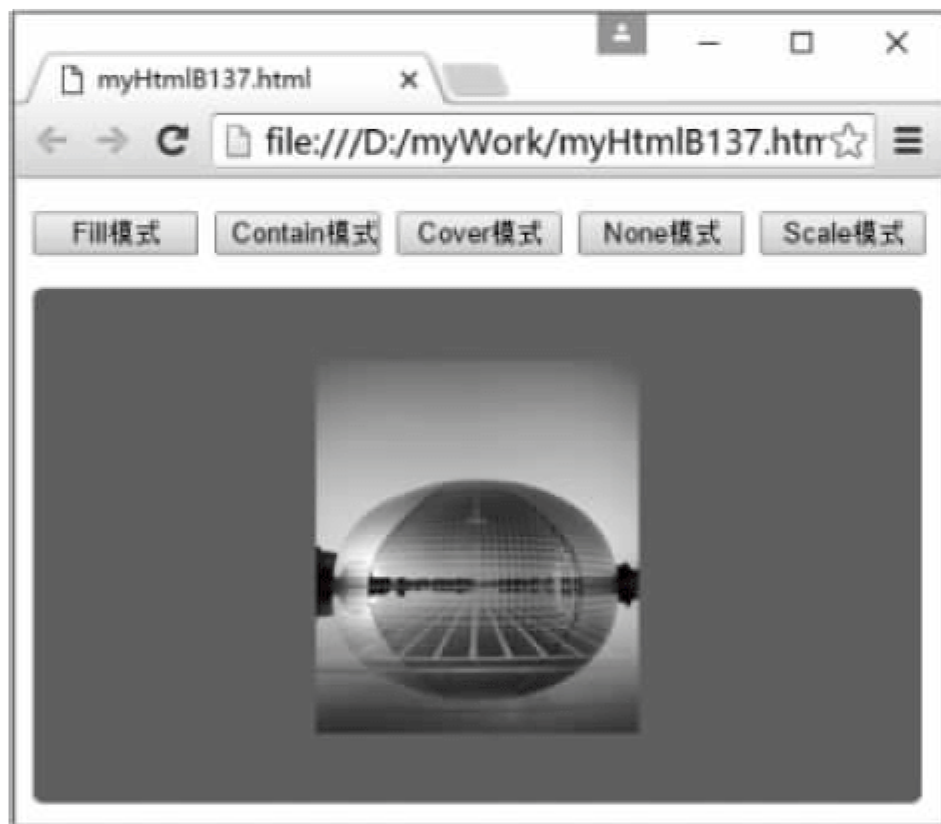


图 537-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
   $(document).ready(function() {
    $("#myBtnFill").click(function() {                 //Fill 模式
     $("img").css("object-fit", "fill");
    });
    $("#myBtnContain").click(function() {              //Contain 模式
     $("img").css("object-fit", "contain");
    });
    $("#myBtnCover").click(function() {                //Cover 模式
     $("img").css("object-fit", "cover");
    });
    $("#myBtnNone").click(function() {                 //None 模式
     $("img").css("object-fit", "none");
    });
    $("#myBtnScaledown").click(function() {            //Scale-down 模式
     $("img").css("object-fit", "scale-down"); });});
 </script>
  <style>
   div { width: 430px; height: 250px; background-color: darkgreen;
         border-radius: 5px; }
   img { width: 100%;  height: 100%; border-radius: 5px; }
   input{width:80px;}
  </style></head>
<body><p><input type="button" value="Fill 模式" id="myBtnFill" />
    <input type="button" value="Contain 模式" id="myBtnContain" />
    <input type="button" value="Cover 模式" id="myBtnCover" />
```

```
        <input type="button" value="None 模式" id="myBtnNone" />
        <input type="button" value="Scale 模式" id="myBtnScaledown" /></p>
    <div><img src="img/B005B.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，object-fit 属性用于设置当前子元素在容器中的适应模式，object-fit 属性包括下列 5 个属性值。

(1) fill：默认值，表示替换内容(子元素)拉伸填满整个容器，不保证保持原有的比例。

(2) contain：表示保持原有尺寸比例，保证替换内容尺寸一定可以在容器里面放得下，因此该属性值可能会在容器内留下空白。

(3) cover：表示保持原有尺寸比例，保证替换内容(子元素)尺寸一定大于容器尺寸，宽度和高度至少有一个和容器一致，因此该属性值可能会让替换内容(如图片)部分区域不可见。

(4) none：表示保持原有尺寸比例，同时保持替换内容原始尺寸大小。

(5) scale-down：中文释义为"降低"，就好像依次设置了 none 或 contain，最终呈现的是尺寸比较小的那个。

此实例的源文件名是 myHtmlB137.html。

538 使用 calc()实现元素与容器的同步变化

此实例主要通过使用 calc()函数根据页面(容器)的当前宽度值动态计算子元素(图片)的宽度，从而实现子元素(图片)与容器(页面)同步变化。当在 Google Chrome 浏览器中显示该页面时，拖动浏览器右端的边框线改变页面大小，则图片的宽度也随之缩放，但图片与页面右端始终相距 50px，如图 538-1 所示。有关此实例的主要代码如下。



图 538-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myDiv{ width:calc(100% - 50px); height:250px;background-image: url(img/B104.jpg); margin: 1px;
          background-repeat: no-repeat; background-size: 100% 100%;border-radius: 5px;}
 </style></head>
<body><div id="myDiv">拖动浏览器边框线改变大小，则图片随之缩放，但与右端始终相距 50px.</div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，width:calc(100% - 50px)用于根据

当前页面的宽度设置元素(图片)的宽度。例如,如果当前容器(页面)的宽度是 400px,在经过 calc(100%－50px)计算之后元素(图片)的宽度就是 350px。在 CSS3 中,calc()函数用于动态计算长度值,任何长度值都可以使用 calc()函数进行计算。calc()函数支持＋、－、*、/、mod 运算,以后还可能加入 min()/max()等运算。

此实例的源文件名是 myHtmlB104.html。

539　使用 column-rule 属性设置列分隔样式

此实例主要通过设置列的 column-rule 属性实现单行配置多列布局的列与列之间的列规则。当在 Google Chrome 浏览器中显示该页面时,以 4 列、双线方式显示的“短歌行”如图 539-1 所示。有关此实例的主要代码如下。



图　539-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myContainer{width: 400px; margin: 10px; background-color: #FFF;
                border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999,
                0 0 40px rgba(0, 0, 0, 0.06) inset; position: relative;
                border-radius: 5px; padding: 20px;text-align: center;}
  /*设置 4 列,以灰色双线分隔*/
  #myColumn{-webkit-column-count:4; -webkit-column-rule: 3px double gray;}
  body{background-color: lightgray;}
 </style></head>
<body><div id="myContainer"><div id="myColumn"><h3>短歌行</h3><p>对酒当歌,人生几何!譬如朝露,
去日苦多。慨当以慷,忧思难忘。何以解忧?唯有杜康。青青子衿,悠悠我心。但为君故,沉吟至今。呦呦鹿鸣,
食野之苹。我有嘉宾,鼓瑟吹笙。明明如月,何时可掇?忧从中来,不可断绝。越陌度阡,枉用相存。契阔谈讌,
心念旧恩。月明星稀,乌鹊南飞。绕树三匝,何枝可依?山不厌高,海不厌深。周公吐哺,天下归心。</p></div>
</div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-column-count:4 表示在 div 块中有 4 列;-webkit-column-rule: 3px double gray 表示列宽为 3px,以灰色双线分隔。column-rule 属性是一个简写属性,用于设置所有 column-rule-* 属性,即设置宽度、样式和颜色规则,column-rule 属性的语法格式如下。

```
column-rule: column-rule-width column-rule-style column-rule-color
```

其中，column-rule-width 表示设置列之间的宽度规则；column- rule-style 表示设置列之间的样式规则，即 none| hidden|dotted|dashed|solid|double| groove|ridge|inset|outset；column-rule-color 表示设置列之间的颜色规则。

此实例的源文件名是 myHtmlB077.html。

540 保持子元素在水平和垂直方向上始终位于正中

此实例主要通过设置元素的 position、top、left 等属性实现保持子元素在水平和垂直方向上始终位于容器的正中。当在 Google Chrome 浏览器中显示该页面时，无论怎样拖动浏览器的边框线改变窗口大小，图片总是位于浏览器的正中，如图 540-1 所示。有关此实例的主要代码如下。



图 540-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <style type = "text/css">
  img{ border - radius: 10px; width: 400px; height:250px; position: absolute; top: calc(50 % - 125px);
left: calc(50 % - 200px); }
 </style></head>
<body><img src = "img/B151B.jpg"/></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，top：calc(50％－125px)表示 img 元素的左上角坐标 top 值是容器(浏览器窗口)高度的 50％减去 img 元素高度(250px)的一半(即 125px)的结果值；left：calc(50％－200px)表示 img 元素的左上角坐标 left 值是容器(浏览器窗口)宽度的 50％减去 img 元素宽度(400px)的一半(即 200px)的结果值，任何长度值都可以使用 calc()函数进行计算；position：absolute 表示 img 元素采取绝对定位模式。在 CSS3 中，position 属性用于检索对象的定位方式，绝对定位的元素，在 top、right、bottom、left 属性未设置时会紧随在其前面的兄弟元素之后，但在位置上不影响常规流中的任何元素。position 属性的语法格式如下。

```
position:static | relative | absolute | fixed | center | page | sticky
```

其中，各属性值的说明如下。

(1) static：表示对象遵循常规流，此时 4 个定位偏移属性不会被应用。

(2) relative：表示对象遵循常规流，并且参照自身在常规流中的位置通过 top、right、bottom、left 这 4 个定位偏移属性进行偏移时不会影响常规流中的任何元素。

(3) absolute：表示对象脱离常规流，此时偏移属性参照的是离自身最近的定位祖先元素，如果没

有定位的祖先元素，则一直回溯到 body 元素。盒子的偏移位置不影响常规流中的任何元素，其 margin 不与其他任何 margin 折叠。

（4）fixed：与 absolute 一致，但偏移定位是以窗口为参考，当出现滚动条时对象不会随着滚动。

（5）center：与 absolute 一致，但偏移定位是以定位祖先元素的中心点为参考。盒子在其包含容器垂直、水平居中。

（6）page：与 absolute 一致。元素在分页媒体或者区域块内，元素的包含块始终是初始包含块，否则取决于每个 absolute 模式。

（7）sticky：表示对象在常态时遵循常规流。它就像是 relative 和 fixed 的合体，当在屏幕中时按常规流排版，当卷动到屏幕外时的表现如 fixed。该属性的表现是现实中大家所见到的吸附效果。

此实例的源文件名是 myHtmlB178.html。

541　在垂直方向上居中显示在一行文本中插入的图像

此实例主要通过设置 vertical-align 属性实现在垂直方向上居中显示在一行文本中插入的图像。当在 Google Chrome 浏览器中显示该页面时，图像居中显示的效果如图 541-1 所示。有关此实例的主要代码如下。



图　541-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  /*创建阴影盒子*/
   div { display: inline-block; padding: 15px; width: 450px; height: 218px; margin: 10px; background-
        color: #FFF; border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0,
        0.06) inset; position: relative; border-radius: 5px; }
   body { background-color: lightgray; }
   img {border-radius: 5px;width: 80px; margin: 5px; height: 100px;
        vertical-align: middle; }  /*在垂直方向上居中显示图像*/
 </style></head>
<body><div><p>阿尔伯特?爱因斯坦<img src="img/B189.jpg"/>(1879.3.14-1955.4.18)犹太裔物理学家。
他于1879年出生于德国乌尔姆市的一个犹太人家庭(父母均为犹太人),1900年毕业于苏黎世联邦理工学院,入
瑞士国籍。1905年,获苏黎世大学哲学博士学位,爱因斯坦提出光子假设,成功解释了光电效应,因此获得1921
年诺贝尔物理奖,同年,创立狭义相对论。1915年创立广义相对论。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，vertical-align：middle 用于居中对齐图像，如果设置为 vertical-align：bottom，则在底部对齐图像。在 CSS 中，vertical-align 属性用于设置

元素的垂直对齐方式，该属性定义行内元素的基线相对于该元素所在行的基线垂直对齐，并且允许指定负长度值和百分比值，但这会使元素降低而不是升高。在单元格中，这个属性用于设置单元格内容的对齐方式。vertical-align 属性支持的属性值的说明如下。

（1）baseline：默认值，表示元素放置在父元素的基线上。

（2）sub：表示垂直对齐文本的下标。

（3）super：表示垂直对齐文本的上标。

（4）top：表示把元素的顶端与行中最高元素的顶端对齐。

（5）text-top：表示把元素的顶端与父元素字体的顶端对齐。

（6）middle：表示把此元素放置在父元素的中部。

（7）bottom：表示把元素的顶端与行中最低的元素的顶端对齐。

（8）text-bottom：表示把元素的底端与父元素字体的底端对齐。

（9）length：表示具体数字。

（10）%：表示使用 line-height 属性的百分比值来排列此元素，允许使用负值。

（11）inherit：规定应该从父元素继承 vertical-align 属性的值。

此实例的源文件名是 myHtmlB189.html。

542　等间距排列一行中的各个图像（元素）

此实例主要设置容器的 text-align-last 属性为 justify，从而实现均匀分布单行中的多个独立的图像。当在 Google Chrome 浏览器中显示该页面时，单击"两端对齐"按钮，则在容器中均匀分布 4 个独立的图像，如图 542-1 所示；单击"向左对齐"按钮，则 4 个独立的图像将靠在容器的左侧，如图 542-2 所示；单击"向右对齐"按钮，则 4 个独立的图像将靠在容器的右侧，如图 542-3 所示；单击"居中对齐"按钮，则 4 个独立的图像将在容器中居中对齐，如图 542-4 所示。有关此实例的主要代码如下。

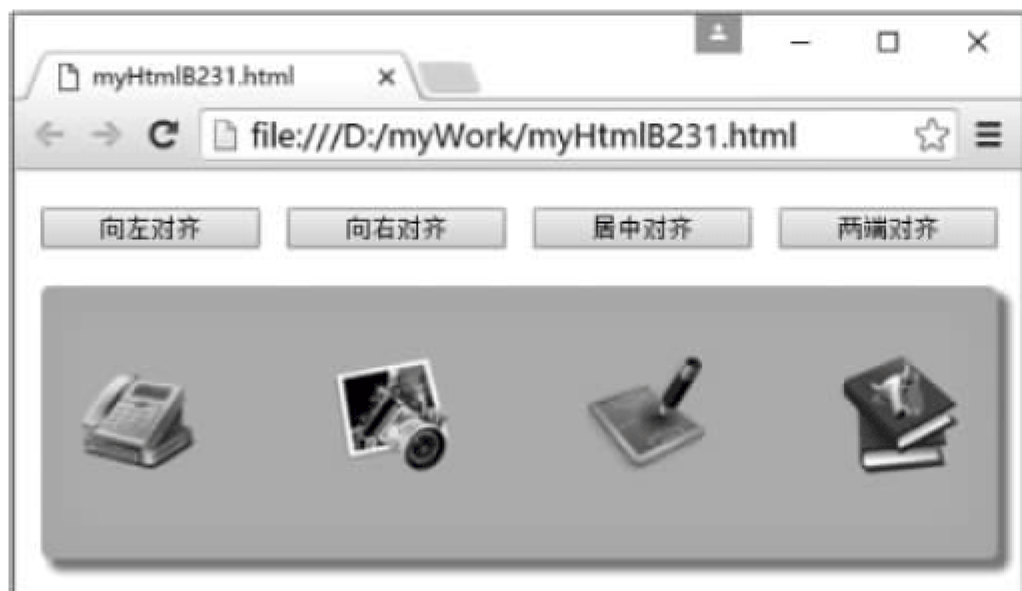


图　542-1



图　542-2



图　542-3



图　542-4

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myLeft").click(function() {              //向左对齐
    $("div").css("text-align-last", "left");
   });
   $("#myRight").click(function() {             //向右对齐
    $("div").css("text-align-last", "right");
   });
   $("#myCenter").click(function() {            //居中对齐
    $("div").css("text-align-last", "center");
   });
   $("#myJustify").click(function() {           //两端对齐
    $("div").css("text-align-last", "justify"); });});
 </script>
 <style>
   div { box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06) inset;
         height: 100px; width: 450px; text-align-last: justify; padding: 20px; border-radius: 5px;
         background-color: aquamarine;}
   img{ width:60px; height:60px; }
   input{ width:112px; margin: 3px;}
 </style></head>
<body><center><p><input type="button" value="向左对齐" id="myLeft"/>
    <input type="button" value="向右对齐" id="myRight"/>
    <input type="button" value="居中对齐" id="myCenter"/>
    <input type="button" value="两端对齐" id="myJustify"/></p>
 <div><p><img src="img/B066A.png"/><img src="img/B067A.png"/>
         <img src="img/B209A.png"/><img src="img/B209B.png"/></p></div>
</center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，text-align-last：justify 用于均匀分布容器中的子元素。在 CSS 中，text-align-last 属性规定如何对齐文本的最后一行，当将独立的图像视作单个汉字之后也能实现类似的效果。text-align-last 属性的语法格式如下。

```
text-align-last: auto | left | right | center | justify | start | end | initial | inherit
```

其中，各属性值的意义如下。

(1) auto：该属性值是默认值，用于使最后一行被调整，并向左对齐。

(2) left：该属性值用于使最后一行向左对齐。

(3) right：该属性值用于使最后一行向右对齐。

(4) center：该属性值用于使最后一行居中对齐。

(5) justify：该属性值用于使最后一行被调整为两端对齐。

(6) start：该属性值用于使最后一行在行开头对齐(如果 text-direction 属性值是从左到右，则向左对齐；如果 text-direction 属性值是从右到左，则向右对齐)。

(7) end：该属性值用于使最后一行在行末尾对齐(如果 text-direction 属性值是从左到右，则向右对齐；如果 text-direction 属性值是从右到左，则向左对齐)。

(8) initial：设置该属性为它的默认值。

(9) inherit：表示从父元素继承该属性。

此实例的源文件名是 myHtmlB231.html。

543 调整多个元素在垂直方向上的间隔距离

此实例主要设置文本行间距的 line-height 属性，从而调整多个图像元素在垂直方向上的间隔距离。当在 Google Chrome 浏览器中显示该页面时，单击"小间距显示"按钮，则 3 幅图像在垂直方向上的显示效果如图 543-1 所示；单击"大间距显示"按钮，则 3 幅图像在垂直方向上的显示效果如图 543-2 所示。有关此实例的主要代码如下。

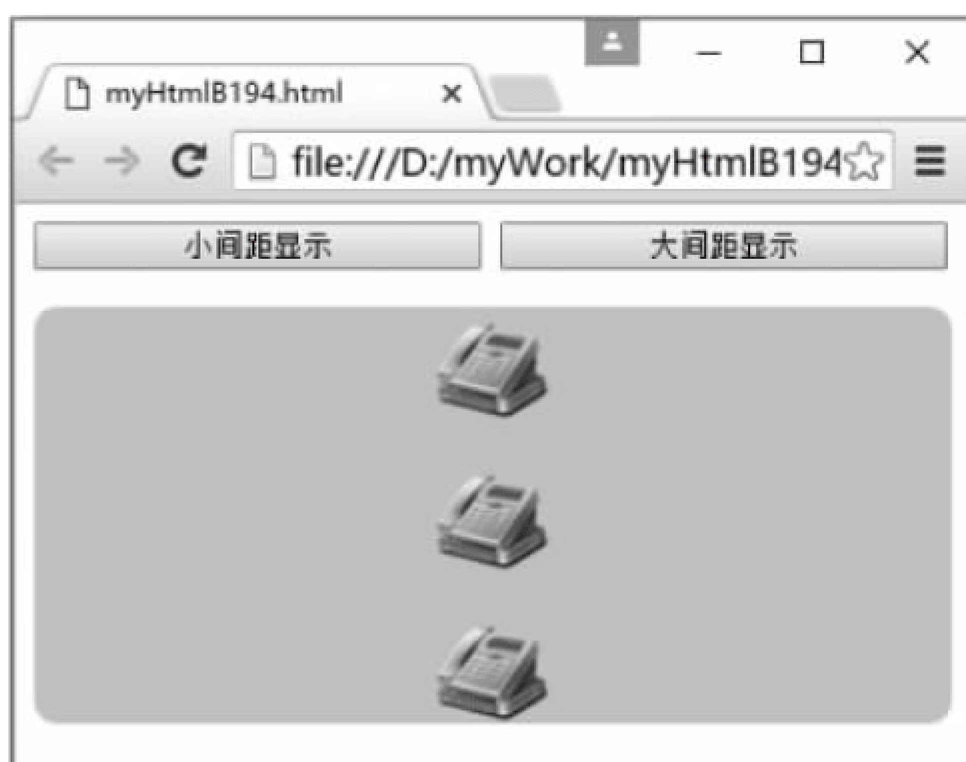


图 543-1



图 543-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnSmall").click(function() {          //小间距显示
    $("p").css("line-height", "5px");
   });
   $("#myBtnBig").click(function() {            //大间距显示
    $("p").css("line-height", "100px"); });});
 </script>
 <style type="text/css">
  div { text-align: center;background-color: lightgray;
       width:400px; border-radius: 10px; }
  img{ width:50px; height:50px; }
  button{width:195px;}
 </style></head>
<body><span><button id="myBtnSmall"/>小间距显示</button>
      <button id="myBtnBig"/>大间距显示</button></span>
<div><p><img src="img/B066A.png"></p><p><img src="img/B066A.png"></p>
     <p><img src="img/B066A.png"></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，line-height 属性用于设置<p>标签内部的文本及其之间的距离(行高)，该属性会影响行框的布局，在应用到一个块级元素时它定义了该

元素中基线之间的最小距离而不是最大距离。line-height 与 font-size 的计算值之差(在 CSS 中成为“行间距”)分为两半，分别加到一个文本行内容的顶部和底部，可以包含这些内容的最小框就是行框。line-height 属性支持的属性值说明如下。

(1) normal：默认值，设置合理的行间距。

(2) number：设置数字，此数字会与当前的字体尺寸相乘来设置行间距。

(3) length：设置固定的行间距。

(4) %：基于当前字体尺寸的百分比行间距。

(5) inherit：规定应该从父元素继承 line-height 属性的值。

此实例的源文件名是 myHtmlB194.html。

544　自定义子元素在水平方向上的排列方向

此实例主要通过设置元素的 direction 属性实现动态重置子元素在水平方向上的排列方向。当在 Google Chrome 浏览器中显示该页面时，单击“从右至左排列”按钮，则 3 幅图像的排列方向如图 544-1 所示；单击“从左至右排列”按钮，则 3 幅图像的排列方向如图 544-2 所示。有关此实例的主要代码如下。



图　544-1



图　544-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnRTL").click(function() {      //从右至左排列
    $("#myContainer").css("direction", "rtl");
   });
   $("#myBtnLTR").click(function() {     //从左至右排列
    $("#myContainer").css("direction", "ltr"); });});
 </script>
 <style type="text/css">
   #myContainer {display: -webkit-flex; width: 450px; }
   .myView {width: 130px; padding: 10px; background-color: darkgreen;
           margin: 1px; text-align: center; border-radius: 5px;}
   img{ border-radius: 5px;}
 </style></head>
<body>
<p><input type="button" value="从右至左排列" id="myBtnRTL" style="width:220px"/>
```

```
<input type="button" value="从左至右排列" id="myBtnLTR" style="width:220px"/></p>
<div id="myContainer"><div class="myView" ><img src=" img/B102A.jpg"></div>
 <div class="myView"><img src="img/B102B.jpg"></div>
 <div class="myView"><img src="img/B102C.jpg"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，direction 属性规定元素的方向，该属性主要有 ltr 和 rtl 两个属性值。ltr 是默认值，表示 left-to-right，就是从左往右的意思，比方说前后两个图片，在默认情况下，DOM 在前的就显示在左边；rtl 则是 right-to-left 的缩写，就是从右往左的意思，在默认情况下，DOM 在前的就显示在右侧，而且是在容器的右端。

此实例的源文件名是 myHtmlB135.html。

545 允许或禁止与相邻同级元素的 float 关系

此实例主要通过设置元素的 clear 属性实现允许或禁止当前元素与相邻同级元素的 float 关系。当在 Google Chrome 浏览器中显示该页面时，单击“文字在下端”按钮，则将禁止文本的 left 属性值，即禁止向左流式布局，因此文本在下端，如图 545-1 所示；单击“文字在右下端”按钮，则将取消禁止文本的 left 属性值，即允许向左流式布局(此属性是图像的 float 属性值)，因此文本在右下端，如图 545-2 所示。有关此实例的主要代码如下。



图 545-1



图 545-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnBottom").click(function() {          //文字在下端
    $("p").css("clear", "left");
  });
   $("#myBtnRight").click(function() {           //文字在右下端
    $("p").css("clear", "none"); });});
 </script>
 <style type="text/css">
  .box { width: 400px;border-radius: 5px; padding: 3px;
        background-color: aqua;}
```

```
    img {float: left;margin: 5px;border-radius: 5px;}
    input {width: 195px;border-radius: 2px; padding: 3px;margin: 2px;}
    p{ margin: 2px;font-size: 14px;text-indent: 25px;}
  </style></head>
<body><div><input type="button" value="文字在下端" id="myBtnBottom"/>
      <input type="button" value="文字在右下端" id="myBtnRight"/></div>
<div class="box"><img src="img/B102B.jpg"><p>中央电视台(简称央视,英语 China Central Television,简称 CCTV),是中华人民共和国国家电视台。1958年5月1日试播,9月2日正式播出。初名北京电视台,1978年5月1日更名为中央电视台。中央电视台是中国重要的新闻舆论机构,是党、政府和人民的重要喉舌,是中国重要的思想文化阵地,是当今中国最具竞争力的主流媒体之一,具有传播新闻、社会教育、文化娱乐、信息服务等多种功能,是全国公众获取信息的主要渠道,也是中国了解世界、世界了解中国的重要窗口,在国际上的影响正日益增强。</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,clear 属性规定元素的哪一侧不允许其他浮动(float)元素。在 CSS1 和 CSS2 中,这是通过自动为清除元素(即设置了 clear 属性的元素)增加上外边距实现的。在 CSS2.1 中,会在元素上外边距之上增加清除空间,而外边距本身并不改变。不论哪一种改变,最终结果都一样,如果声明为左边或右边清除,会使元素的上外边框边界刚好在该边上浮动元素的下外边距边界之下。clear 属性支持的属性值的意义如下。

(1) left:表示在左侧不允许浮动元素。

(2) right:表示在右侧不允许浮动元素。

(3) both:表示在左、右两侧均不允许浮动元素。

(4) none:默认值,表示允许浮动元素出现在两侧。

(5) inherit:表示规定应该从父元素继承 clear 属性的值。

此实例的源文件名是 myHtmlB141.html。

546　使用多列布局实现元素内容的分多列显示

此实例主要通过设置 column-count 和 column-width 属性实现以多列布局的方式将一个元素的内容分为两列或多列显示。当在 Google Chrome 浏览器中显示该页面时,以两列方式显示的“长恨歌”如图 546-1 所示。有关此实例的主要代码如下。



图　546-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myContainer { width: 560px;}
  #myColumn { -webkit-column-count: 2;  /*设置容器有两列,列宽为270px*/
              -webkit-column-width: 270px;text-align: center; }
  /*创建多级辉光的标题文字*/
  #myTitle { width: 560px; height: 100px; color: black; text-shadow: 0 0 5px #FFF, 0 0 10px #FFF, 0 0
           15px #FFF, 0 0 30px #FF00DE, 0 0 45px #FF00DE; font: bold 40px/100% "微软雅黑",
           "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans; display: table-cell; vertical-
           align: middle; text-align: center;}
  img {margin: 5px;border-radius: 5px;}
 </style></head>
<body><div id="myContainer"><div id="myTitle">中华古诗词经典欣赏</div>
 <div id="myColumn"><h3>长歌行</h3><p>青青园中葵,朝露待日晞。<br>
     阳春布德泽,万物生光辉。<br>常恐秋节至,焜黄华叶衰。<br>
     百川东到海,何时复西归?<br>少壮不努力,老大徒伤悲。<br></p>
  <h3>长恨歌</h3><p>汉皇重色思倾国,御宇多年求不得。<br>
     杨家有女初长成,养在深闺人未识。<br>天生丽质难自弃,一朝选在君王侧。<br>
     回眸一笑百媚生,六宫粉黛无颜色。<br> 春寒赐浴华清池,温泉水滑洗凝脂。<br>
     侍儿扶起娇无力,始是新承恩泽时。<br>云鬓花颜金步摇,芙蓉帐暖度春宵。<br>
    春宵苦短日高起,从此君王不早朝。<br></p><img src="img/B074.jpg"></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,column-count 属性用于规定元素应该被划分的列数,该属性的语法格式如下。

```
column-count: number | auto
```

其中,number 属性值表示元素内容将被划分的最佳列数,即一具体数字;auto 属性值表示由其他属性决定列数,例如 column-width。

column-width 属性用于规定列的宽度,该属性的语法格式如下。

```
column-width: auto | length
```

其中,auto 属性值表示由浏览器决定列宽;length 属性值规定列的宽度,即一具体数字。

此实例的源文件名是 myHtmlB074.html。

547 自定义多列布局的列与列之间的分隔样式

此实例主要通过设置列的 column-count、column-rule-style、column-rule-width、column-rule-color 等属性值实现定制多列布局的列与列之间的分隔样式。当在 Google Chrome 浏览器中显示该页面时,以 3 列、虚线方式显示的"长恨歌节选"如图 547-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  #myContainer { width: 420px; margin: 10px;background-color: #FFF;
                 border: 1px solid #EEE; box-shadow: 5px 5px 5px 1px #999,
                 0 0 40px rgba(0, 0, 0, 0.06) inset;position: relative;
                 border-radius: 5px; padding: 20px;text-align: center;}
  /*设置3列,以黑色虚线分隔*/
```

```
  #myColumn { - webkit - column - count:3;  - webkit - column - rule - style:dotted;  - webkit - column -
            rule - width:2px;  - webkit - column - rule - color:black;}
  body { background - color: lightgray;}
 </style></head>
<body><div id = "myContainer"><div id = "myColumn"><h3>长恨歌节选</h3>
  <p>金阙西厢叩玉扃,转教小玉报双成。闻道汉家天子使,九华帐里梦魂惊。揽衣推枕起徘徊,珠箔银屏迤逦开。云鬓半偏新睡觉,花冠不整下堂来。风吹仙袂飘飖举,犹似霓裳羽衣舞。玉容寂寞泪阑干,梨花一枝春带雨。含情凝睇谢君王,一别音容两渺茫。昭阳殿里恩爱绝,蓬莱宫中日月长。回头下望人寰处,不见长安见尘雾。惟将旧物表深情,钿合金钗寄将去。钗留一股合一扇,钗擘黄金合分钿。但教心似金钿坚,天上人间会相见。临别殷勤重寄词,词中有誓两心知。七月七日长生殿,夜半无人私语时。在天愿作比翼鸟,在地愿为连理枝。天长地久有时尽,此恨绵绵无绝期。</p></div></div></body></html>
```



图 547-1

上面有底纹的代码是此实例的核心代码。在该部分代码中,-webkit-column-count:3 表示 div 块中有 3 列; -webkit-column-rule-width:2px 表示 div 块中的列分隔线的宽度是 2px; -webkit-column-rule-color:black 表示 div 块中的列分隔线的颜色是黑色; -webkit- column-rule-style:dotted 表示 div 块中的列分隔线是虚线。column-rule-style 属性的语法格式如下。

```
column - rule - style: none | hidden | dotted | dashed | solid | double
```

其中,none 定义没有规则、hidden 定义隐藏规则、dotted 定义点状规则、dashed 定义虚线规则、solid 定义实线规则、double 定义双线规则。

此外,在设置列与列之间的样式时也经常用到 column-gap 属性,column-gap 属性规定列之间的间隔,如果列之间设置了 column-rule,它会在间隔中间显示。column-gap 属性的语法格式如下。

```
column - gap: length | normal
```

其中,属性值 length 表示把列间的间隔设置为指定的长度,例如 50px; normal 规定列间的间隔为一个常规的间隔,W3C 建议的值是 1em。

此实例的源文件名是 myHtmlB075.html。

548 使用盒布局解决多列底部不能对齐的问题

此实例主要设置容器的 display 属性为 box,从而实现使用盒布局解决多列底部不能对齐的问题。当在 Google Chrome 浏览器中显示该页面时,3 列的高度原来是不一致(即根据文本内容的大小来显

示列的高度)的,但是在设置容器的 display 属性为 box 后 3 列的高度与最长列的高度一致,如图 548-1 所示。有关此实例的主要代码如下。

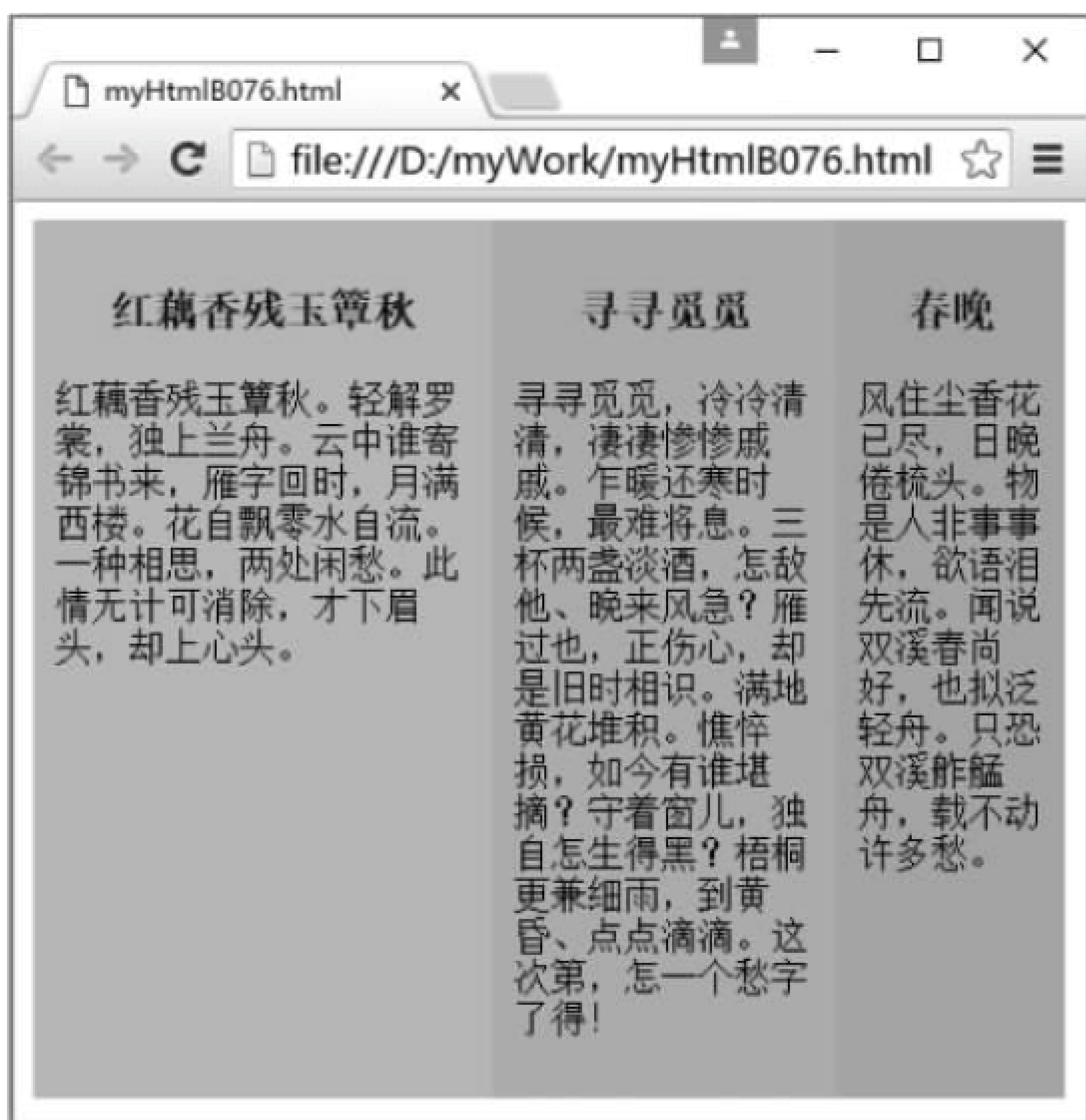


图 548-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   #myContainer{ display: -webkit-box; }
   #myLeft{width: 200px;padding: 10px; background-color: lightblue;}
   #myMiddle{ width: 150px; padding: 10px; background-color: lightpink; }
   #myRight{ width: 100px; padding: 10px; background-color: lightgreen; }
   #myLeft, #myMiddle, #myRight{ box-sizing: border-box; }
  h3{text-align: center; }
 </style></head>
<body><div id="myContainer">
 <div id="myLeft"><h3>红藕香残玉簟秋</h3><p>红藕香残玉簟秋。轻解罗裳,独上兰舟。云中谁寄锦书来,雁字回时,月满西楼。花自飘零水自流。一种相思,两处闲愁。此情无计可消除,才下眉头,却上心头。</p>
</div>
 <div id="myMiddle"><h3>寻寻觅觅</h3><p>寻寻觅觅,冷冷清清,凄凄惨惨戚戚。乍暖还寒时候,最难将息。三杯两盏淡酒,怎敌他、晚来风急?雁过也,正伤心,却是旧时相识。满地黄花堆积。憔悴损,如今有谁堪摘?守着窗儿,独自怎生得黑?梧桐更兼细雨,到黄昏、点点滴滴。这次第,怎一个愁字了得!</p></div>
 <div id="myRight"><h3>春晚</h3><p>风住尘香花已尽,日晚倦梳头。物是人非事事休,欲语泪先流。闻说双溪春尚好,也拟泛轻舟。只恐双溪舴艋舟,载不动许多愁。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,display:-webkit-box 表示将对象作为弹性伸缩盒显示。display 属性的语法格式如下。

```
display:none | inline | block | list-item | inline-block | table | inline-table | table-caption | table-cell | table-row | table-row-group | table-column | table-column-group | table-footer-group | table-header-group | run-in | box | inline-box | flexbox | inline-flexbox | flex | inline-flex
```

其中,各属性值的说明如下。

(1) none:表示隐藏对象,与 visibility 属性的 hidden 值不同,它不为被隐藏的对象保留其物理空间。

（2）inline：指定对象为内联元素。
（3）block：指定对象为块元素。
（4）list-item：指定对象为列表项目。
（5）inline-block：指定对象为内联块元素。
（6）table：指定对象作为块元素级的表格，类同于 html 标签< table >。
（7）inline-table：指定对象作为内联元素级的表格，类同于 html 标签< table >。
（8）table-caption：指定对象作为表格标题，类同于 html 标签< caption >。
（9）table-cell：指定对象作为表格单元格，类同于 html 标签< td >。
（10）table-row：指定对象作为表格行，类同于 html 标签< tr >。
（11）table-row-group：指定对象作为表格行组，类同于 html 标签< tbody >。
（12）table-column：指定对象作为表格列，类同于 html 标签< col >。
（13）table-column-group：指定对象作为表格列组显示，类同于 html 标签< colgroup >。
（14）table-header-group：指定对象作为表格标题组，类同于 html 标签< thead >。
（15）table-footer-group：指定对象作为表格脚注组，类同于 html 标签< tfoot >。
（16）run-in：根据上下文决定对象是内联对象还是块级对象。
（17）box：将对象作为弹性伸缩盒显示。
（18）inline-box：将对象作为内联块级弹性伸缩盒显示。
（19）flexbox：将对象作为弹性伸缩盒显示。
（20）inline-flexbox：将对象作为内联块级弹性伸缩盒显示。
（21）flex：将对象作为弹性伸缩盒显示。
（22）inline-flex：将对象作为内联块级弹性伸缩盒显示。
此实例的源文件名是 myHtmlB076.html。

549　在自适应多列布局中决定是否开启新列

此实例主要通过设置 column-break-before 属性值实现在自适应的多列布局中决定是否开启新列。当在 Google Chrome 浏览器中显示该页面时，单击"新建一列"按钮，则"黄鹤楼"将从第 3 列开始显示，如图 549-1 所示；单击"接续前列"按钮，则"黄鹤楼"将在第 2 列的尾部开始显示，如图 549-2 所示。有关此实例的主要代码如下。



图　549-1



图　549-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnAlways").click(function() {       //新建一列
    $("#myNewColumn").css("column-break-before", "always");
   });
   $("#myBtnAuto").click(function() {         //接续前列
    $("#myNewColumn").css("column-break-before", "auto"); });});
 </script>
 <style type="text/css">
   #myContainer {width: 420px; margin: 10px; background-color: #FFF; border: 1px solid #EEE;
                position: relative; box-shadow: 5px 5px 5px 1px #999, 0 0 40px rgba(0, 0, 0, 0.06)
                inset; border-radius: 5px;padding: 20px; text-align: center;}
  /*设置3列,以黑色虚线分隔*/
  #myColumn {-webkit-column-count: 3; -webkit-column-rule-style: dotted;
             -webkit-column-rule-width: 2px; -webkit-column-rule-color: black;}
  body {background-color: lightgray;}
  input{width:100px}
 </style></head>
<body><div id="myContainer">
 <p><input type="button" value="新建一列" id="myBtnAlways"/>
    <input type="button" value="接续前列" id="myBtnAuto"/></p>
 <div id="myColumn">
  <h3>长恨歌节选</h3><p>回头下望人寰处,不见长安见尘雾。惟将旧物表深情,钿合金钗寄将去。钗留一
股合一扇,钗擘黄金合分钿。但教心似金钿坚,天上人间会相见。临别殷勤重寄词,词中有誓两心知。七月七日
长生殿,夜半无人私语时。在天愿作比翼鸟,在地愿为连理枝。天长地久有时尽,此恨绵绵无绝期。</p>
  <h3 id="myNewColumn">黄鹤楼</h3><p>昔人已乘黄鹤去,此地空余黄鹤楼。黄鹤一去不复返,白云千载空
悠悠。晴川历历汉阳树,芳草萋萋鹦鹉洲。日暮乡关何处是?烟波江上使人愁。</p></div></div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("#myNewColumn").css("column-break-before","always")用于设置当前标题"黄鹤楼"在新的一列中显示。在CSS3中,column-break-before属性用于设置或检索对象之前是否中断,该属性的语法格式如下。

```
column-break-before:auto | always | avoid
```

其中,auto表示既不强迫也不禁止在元素之前中断并产生新列;always表示总是在元素之前中断并产生新列;avoid表示避免在元素之前中断并产生新列。

此实例的源文件名是myHtmlB084.html。

550 在自适应多列布局中决定是否强制开启新列

此实例主要通过设置column-break-after属性值实现在自适应的多列布局中决定后续内容是否开启新列。当在Google Chrome浏览器中显示该页面时,单击"强制后面的内容新建一列"按钮,则"丰都鬼城"将在第2列的开始位置显示,如图550-1所示;单击"允许后面的内容共存一列"按钮,则"丰都鬼城"将在第2列的中部位置显示,与"长寿湖"的简介共存一列,如图550-2所示。有关此实例的主要代码如下。



图 550-1



图 550-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnAlways").click(function() {        //强制后面的内容新建一列
    $("p").css("column-break-after", "always"); });
   $("#myBtnAuto").click(function() {          //允许后面的内容共存一列
    $("p").css("column-break-after", "auto"); });});
 </script>
 <style type="text/css">
  #myContainer {width: 420px;margin: 10px;background-color: #FFF; border: 1px solid #EEE; text-
                align: center; position: relative;border-radius: 5px;padding: 20px; box-shadow: 5px
                5px 5px 1px #999,0 0 40px rgba(0, 0, 0, 0.06) inset; }
  #myColumn {-webkit-column-count: 3;}
  body {background-color: lightgray;}
  input {width: 195px; margin: 5px;}
  img {width: 125px;height: 80px;border-radius: 5px;}
  p {text-align: left; -webkit-column-break-after: always;
     background-color: lightcyan;border-radius: 3px;padding: 5px;}
 </style></head>
<body><div id="myContainer">
  <input type="button" value="强制后面的内容新建一列" id="myBtnAlways"/>
  <input type="button" value="允许后面的内容共存一列" id="myBtnAuto"/>
  <div id="myColumn"><h3>长寿湖</h3><img src="img/B085A.jpg"/><p>长寿湖位于长寿区东部,距重庆
市区 128 千米,为 20 世纪 50 年代拦截龙溪河而成的人工湖,因地处长寿区境内而得名。长寿湖坝长 1014 米,高
51 米,顶宽 8 米,湖水面积 6000 公顷。一般水深 15 米,最深处 50 米。湖内港汊纵横交错,有岛屿 200 多个,是
重庆市最大的湖泊旅游风景区。</p>
```

```
<h3>丰都鬼城</h3><img src="img/B085B.jpg"/><p>丰都鬼城旧名酆都鬼城,古为"巴子别都",东汉和帝永元二年置县,距今已有近 2000 年的历史,位于重庆市下游丰都县的长江北岸,是长江游轮旅客的一个观光胜地。丰都鬼城又称为"幽都"、"鬼国京都"、"中国神曲之乡"。鬼城以各种阴曹地府的建筑和造型而著名。鬼城内有哼哈祠、天子殿、奈河桥、黄泉路、望乡台、药王殿等多座表现阴间的建筑。</p></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("p").css("column-break- after","always")用于设置每个段落(如风景区简介)后面的内容在新的一列中显示。在 CSS3 中,column-break-after 属性用于设置或检索对象之后是否中断,该属性的语法格式如下。

```
column-break-after:auto | always | avoid
```

其中,auto 表示既不强迫也不禁止在元素之后中断并产生新列;always 表示总是在元素之后中断并产生新列;avoid 表示避免在元素之后中断并产生新列。

此实例的源文件名是 myHtmlB085.html。

551 使用随机数模拟照片墙的多图散列布局

此实例主要通过使用 Math.random()函数产生随机数设置图像的坐标、转角、索引等参数,从而实现模拟照片墙的多图散列布局效果。当在 Google Chrome 浏览器中显示该页面时,9 幅图像的散列布局效果如图 551-1 所示;重新刷新当前页面,则 9 幅图像的散列布局效果如图 551-2 所示,即每次显示页面时 9 幅图像的散列布局都各不相同。有关此实例的主要代码如下。



图 551-1



图 551-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
```

```
    $ (function(){
     for(var i = 1;i < 10;i++){
       var x = Math.floor(Math.random() * (document.body.scrollWidth - 170));
       var y = Math.floor(Math.random() * (document.body.scrollHeight - 50));
       var myIndex = Math.floor(Math.random() * 999); //随机 z - index 值
       var myDegree = Math.floor(Math.random() * 360);
        $ (".myBox").append("< img src = 'img/B333A" + i + ".jpg' id = 'img" + i + "' style = 'top:" + y + "px;
left:" + x + "px;z - index:" + myIndex + "; - webkit - transform:rotate(" + myDegree + "deg)'/>"); } });
 </script>
 < style type = "text/css">
  .myBox img { position: absolute; width:120px; height:160px;
               border - radius: 5px; box - shadow: 2px 2px 5px black; }
 </style></head>
< body >< div class = "myBox"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，Math.random()函数用于返回 0 和 1 之间的伪随机数，因此实例通过使用该伪随机数乘以适当的系数作为图像的坐标、转角、Z 轴索引；Math.floor(x)函数用于返回小于等于数字参数的最大整数，即对数字进行下舍入，例如 Math.floor(5.99)的结果是 5 而不是 6。

此实例的源文件名是 myHtmlB333.html。

552　使用随机数模拟瀑布流风格的多图布局

此实例主要通过使用 Math.random()函数产生随机数设置图像的高度，从而实现模拟瀑布流风格对多幅图像进行布局。当在 Google Chrome 浏览器中显示该页面时，以瀑布流风格显示的多幅图像如图 552-1 所示。有关此实例的主要代码如下。

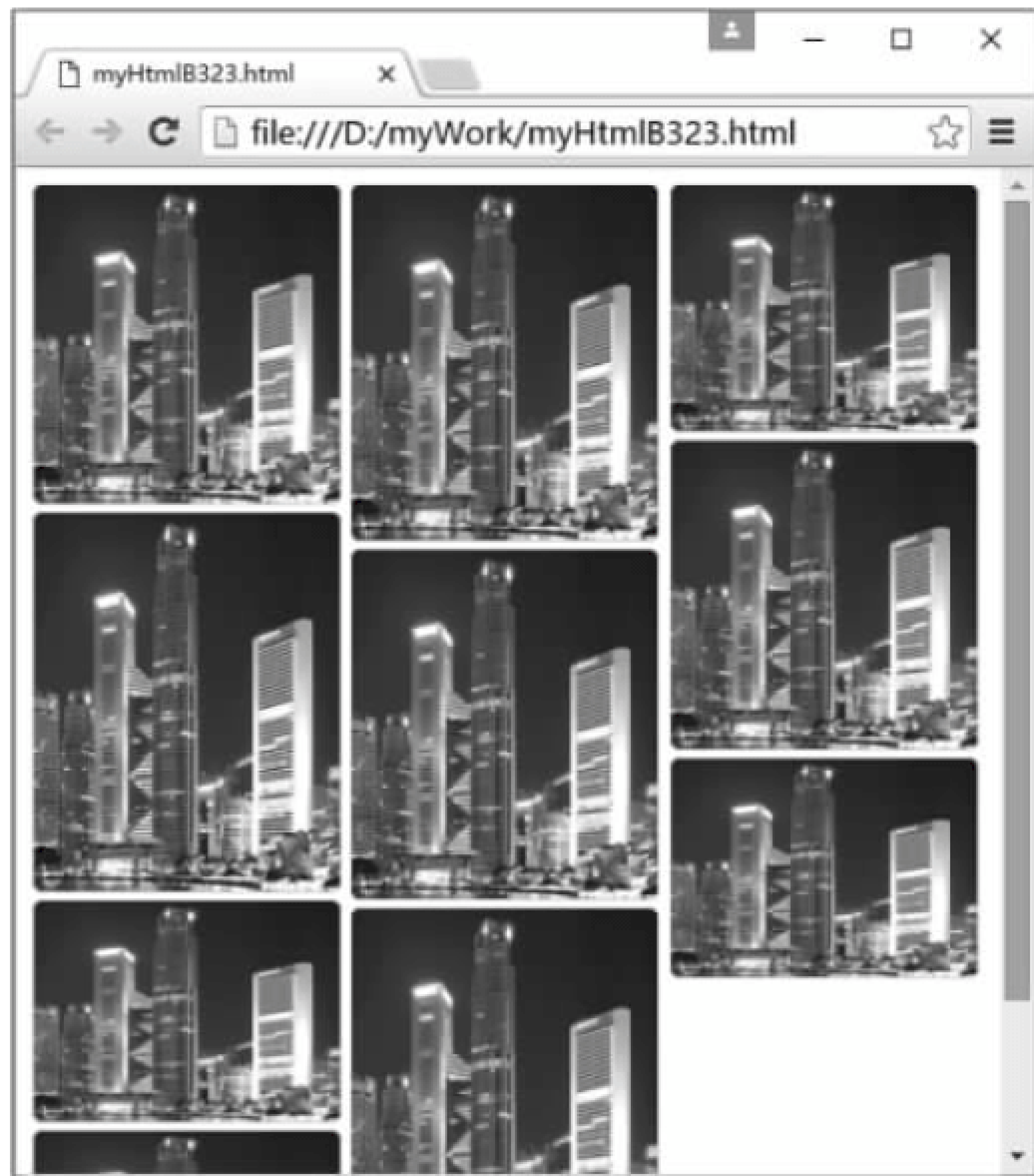


图　552-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {//添加 10 幅图像,并用随机数设置图像高度
   for (var i=0; i<10; i++) {
    $(".myBox").append('<div class="myImage" style="height:'+(Math.floor(Math. random()*100)+
100)+'px;"></div>');  } });
 </script>
 <style type="text/css">
  .myBox { -webkit-column-width: 160px;   -webkit-column-gap: 1px; }
  /*设置图像盒子样式*/
  div:not(.myBox) { -webkit-border-radius: 5px; background: #D9D9D9; border: #CCC 1px solid;
                   display: inline-block; width: 157px; position: relative;}
  /*设置图像样式*/
  div.myImage { background-image: url("img/B3221.jpg");
               background-size: 157px 100%; }
 </style></head>
<body><div class="myBox"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，Math. random()函数返回 0 和 1 之间的伪随机数，因此实例通过使用该伪随机数乘以 100 作为图像高度的波动范围；Math. floor(x)函数返回小于等于数字参数的最大整数，即对数字进行下舍入，例如 Math. floor(5. 99)的结果是 5 而不是 6。

此实例的源文件名是 myHtmlB323. html。

553　通过调整列宽将横向文字变为纵向文字

此实例主要通过设置元素的 column-width 属性实现将横向排列的文字改为纵向排列。当在 Google Chrome 浏览器中显示该页面时，单击“沿水平方向排列文字”按钮，则下面的文字将沿着水平方向排列，如图 553-1 所示；单击“沿垂直方向排列文字”按钮，则下面的文字将沿着垂直方向排列，如图 553-2 所示。有关此实例的主要代码如下。



图　553-1



图　553-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  $(function() {
   $("#myHorizontal").click(function() {       //沿水平方向排列文字
    $("div.myBox").css("-webkit-column-width", "initial");
   });
   $("#myVertical").click(function() {         //沿垂直方向排列文字
    $("div.myBox").css("-webkit-column-width", "30px"); });});
 </script>
 <style type="text/css">
  * { margin: 0; padding: 0;}
  .myBox { font-size: 18px;color: black; padding: 20px;box-shadow: 0 5px 5px darkgrey; border-radius:
         5px; background-color: lightseagreen; margin: 5px;}
  button { width: 198px; margin: 5px;}
 </style></head>
<body><p><button id="myHorizontal">沿水平方向排列文字</button>
   <button id="myVertical">沿垂直方向排列文字</button></p>
<div class="myBox">最早的编程语言是在电脑发明之后产生的,当时是用来控制提花织布机及自动演奏钢琴
的动作。在电脑领域已发明了上千不同的编程语言,而且每年仍有新的编程语言诞生。很多编程语言需要用指
令方式说明计算的程序,而有些编程语言则属于声明式编程,说明需要的结果,而不说明如何计算。</div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,$("div.myBox").css("-webkit-column-width","initial")表示使用默认值设置元素的文字列宽。在 CSS 中,initial 关键字用于设置 CSS 属性为它的默认值,initial 关键字可用于任何 HTML 元素上的任何 CSS 属性。$("div.myBox").css("-webkit-column-width","30px")用于设置元素的文字列宽为 30px。

此实例的源文件名是 myHtmlB327.html。

第7部分

选择器

554　使用 first-line 选择器定制文本的首行样式

此实例主要通过使用 first-line 选择器实现定制文本的首行样式。当在 Google Chrome 浏览器中显示该页面时，两首唐诗的首行将以红色粗体字显示，如图 554-1 所示。有关此实例的主要代码如下。



图　554-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
/* 创建有阴影的盒子 */
  div.myFrame{float:left; margin:5px; width:150px;padding:10px;border:1px solid #BFBFBF; background
              -color:white;box-shadow:2px 2px 3px gray;}
  p:first-line{ color:red; font-size:18px; font-weight: bold;}
 </style></head>
<body><div class="myFrame">
    <p>昨夜星辰昨夜风,<br> 画楼西畔桂堂东。<br> 身无彩凤双飞翼,<br>
        心有灵犀一点通。<br> 隔座送钩春酒暖,<br> 今曹射覆蜡灯红。<br>
        嗟余听鼓应官去,<br> 走马兰台类转蓬。<br></p></div>
<div class="myFrame">
    <p>来去空言去绝踪,<br> 月斜楼上五更钟。<br> 梦为远别啼难唤,<br>
        书被催成墨未浓。<br> 蜡照半笼金翡翠,<br> 麝熏微度绣芙蓉。<br>
        刘郎已恨蓬山远,<br> 更隔蓬山一万重。<br></p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，p:first-line 是一个伪元素选择器。伪元素选择器并不是针对真正的元素使用的选择器，而是针对 CSS 中已经定义的伪元素使用的选择器。在 CSS 中主要有 4 个伪元素选择器，即 first-line 选择器、first-letter 选择器、before 选择器、after 选择器。选择器的使用方法如下。

```
选择器:伪元素{属性:值}
```

选择器也可以与类配合使用，使用方法如下。

```
选择器.类名:伪元素{属性:值}
```

此实例的源文件名是 myHtmlB007.html。

555　使用 first-letter 选择器定制文本的首字样式

此实例主要通过使用 first-letter 选择器实现定制文本的首字样式。当在 Google Chrome 浏览器中显示该页面时，第一首唐诗的首字将以红色粗体字显示，第二首英文诗的首字母也将以红色粗体字显示，如图 555-1 所示。有关此实例的主要代码如下。



图　555-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  div.myFrame{float:left; margin:5px; width:300px;padding:10px;border:1px solid #BFBFBF; background
             - color:white;box - shadow:2px 2px 3px gray;}
  p:first - letter{color:red;font - size:28px;font - weight: bold;}
 </style></head>
<body><div class = "myFrame">
 <p>昨夜星辰昨夜风,画楼西畔桂堂东。<br>身无彩凤双飞翼,心有灵犀一点通。<br>
    隔座送钩春酒暖,今曹射覆蜡灯红。<br>嗟余听鼓应官去,走马兰台类转蓬。<br></p></div>
<div class = "myFrame"><p>Whenever you need me, I'll be here. Whenever you're in trouble, I'm always
near. Whenever you feel alone, and you think everyone has given up...Reach out for me, and I will give you my
everlasting love</p></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，p:first-letter 是一个伪元素选择器。该选择器的主要功能是设置 p 元素中首字母的样式为红色、粗体，字体尺寸为 28px。

此实例的源文件名是 myHtmlB008.html。

556 使用 before 选择器在元素之前插入内容

此实例主要通过使用 before 选择器和 after 选择器实现在 li 元素的前后分别插入新内容。当在 Google Chrome 浏览器中显示该页面时，将在每种图书的名称前面插入"【"、后面插入"】"，如图 556-1 所示。有关此实例的主要代码如下。



图 556-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
   div.myFrame {float: left; margin: 5px; width: 350px; padding: 15px; border: 1px solid #BFBFBF;
                 background-color: white; box-shadow: 2px 2px 3px gray;}
   li {list-style-type: none;}
   li:before{content:"【"; font-size:18px; color:red;}
   li:after{content:"】"; font-size:18px; color:red;}
 </style></head>
<body><div class="myFrame"><h3>10 月份新上榜计算机图书</h3>
    <ul><li>JavaScript DOM 编程艺术</li><li>利用 Python 进行数据分析</li>
        <li>SRE: Google 运维解密</li><li>Head First 设计模式</li>
        <li>深入理解 Java 虚拟机</li><li>JavaScript 高级程序设计</li>
        <li>第一行代码 Android</li></ul></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:before 是一个伪元素选择器，该选择器用于在 li 元素之前插入一些内容，此内容除了文本以外也可以是图片等其他内容，如果将实例中的 li:before{content:"【";font-size:18px;color: red;}修改为 li:before{content:url(img/B005A.jpg);}，则将在每种书名的前面显示图像。li:after 也是一个伪元素选择器，该选择器用于在 li 元素之后插入一些内容，此内容除了文本以外也可以是图像等其他内容，例如 li:after{content: url(img/B005B.jpg); }。

此实例的源文件名是 myHtmlB009.html。

557　使用 before 选择器在页面顶端添加阴影

此实例主要通过在 before 选择器中设置-webkit-box-shadow 属性实现在页面顶端添加阴影。当在 Google Chrome 浏览器中显示该页面时，在页面的顶端将显示 10px 高的阴影，如图 557-1 所示。有关此实例的主要代码如下。

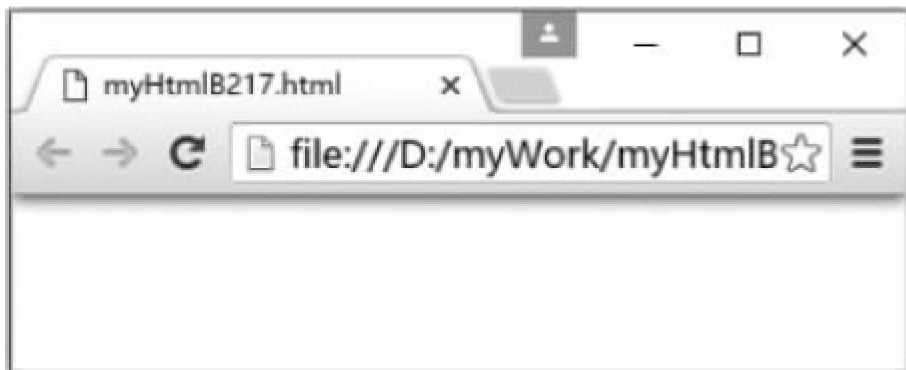


图　557-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body:before { content: "";  position: fixed; top: -10px; left: 0px; width: 100%; height: 10px; -
              webkit-box-shadow: 0px 0px 10px rgba(0,0,0, 0.8); z-index: 999;}
  html { background-color: white; }
 </style></head>
<body></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，body:before 用于设置在页面前显示的内容（此实例即为阴影），常用来和 content 属性一起使用，并且必须定义 content 属性值。content 属性用于插入生成内容，该属性与 before 选择器及 after 选择器配合使用，将生成内容放在一个元素内容的前面或后面。content 属性的语法格式如下。

```
content:normal | string | attr() | uri() | counter()
```

其中，normal 是默认值；string 表示插入文本内容；attr()表示插入元素的属性值；uri()表示插入一个外部资源（如图像、声频、视频或浏览器支持的其他任何资源）；counter()表示一个计数器，用于插入排序标识。

-webkit-box-shadow：0px 0px 10px rgba(0,0,0, 0.8)用于创建高度为 10px、透明度为 0.8 的阴影。

此实例的源文件名是 myHtmlB217.html。

558　使用 before 选择器创建图文并列的按钮

此实例主要通过使用 before 选择器在按钮文本左边插入图像，从而实现创建图像和文本并列显示的按钮。当在 Google Chrome 浏览器中显示该页面时，使用 before 选择器创建的图像和文本并列按钮的效果如图 558-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body { font: 15px/1.3 'Raleway', sans-serif; color: #5E5B64; padding-bottom: 120px; text-align:
       center; }
```

```
      a, a:visited { outline: none; color:black; }
      a:hover { text - decoration: none; }  /* 取消下画线 */
      .myButton { display: inline - block; font - size: 18px; color: black; text - decoration: none !important;
                box - shadow: 0 5px 5px #CCC; padding: 10px 30px; border - radius: 10px;background - image:
                - webkit - linear - gradient(top, snow,red); margin: 5px;  width:250px;}  /* 设置按钮基
                本样式 */
      .myButton:before { display: inline - block; content: ""; margin - right: 15px; background - position:
                    center; background - repeat: no - repeat; background - image: url("img/B324.png");
                    width:16px;height:16px;}  /* 设置按钮左边的图像样式 */
    </style></head>
  <body><a href = "#" class = "myButton">华西包装集团公司</a><br>
  <a href = "#" class = "myButton">兴业投资集团公司</a><br>
  <a href = "#" class = "myButton">重庆市国有资产管理委员会</a><br>
  <a href = "http://www.cq.gov.cn" class = "myButton">重庆市政府网</a><br>
  </body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，before 选择器用于在被选元素的内容前面插入内容，myButton:before{ }即在按钮文本的前面插入内容（图像）。如果需要在按钮文本的后面插入内容（图像），则应该使用 after 选择器。

此实例的源文件名是 myHtmlB324.html。



图 558-1

559 使用 after 选择器创建箭头风格的提示框

此实例主要通过在 after 选择器中以切分边框线的策略切出一个三角形作为指示箭头，从而实现创建箭头风格的提示框。当在 Google Chrome 浏览器中显示该页面时，带三角形箭头提示框的显示效果如图 559-1 所示。有关此实例的主要代码如下。

```
  <!doctype html><html><head><meta charset = "UTF - 8">
   <style type = "text/css">
     * { margin: 0px; padding: 0px;}
    .myTips { position: absolute; top: 50px; left: 50%;transform: translate( - 50%, - 50%);background
             - color:lightgreen; border - radius: 5px; width: 400px; padding: 10px;text - align: left;
             font - size: 14px; }  /* 设置提示框的圆角矩形 */
    /* 在圆角矩形的下面添加三角形(即指示箭头) */
```

```
  . myTips: after { content: ' ';   position: absolute; bottom: 0; left: 50%; border: 20px solid
                    transparent; border-top-color:lightgreen;border-bottom: 0;  border-left: 0;
                    margin: 0 0 -20px -10px; }
  .myBox{ margin: 105px auto 10px; width: 420px; height: 250px; border-radius: 5px;
          box-shadow: 2px 2px 8px black; }
  img{ width: 420px; height: 250px; border-radius: 5px; }
 </style></head>
<body><div class="myTips">香港是全球高度繁荣的国际大都会之一,全境由香港岛、九龙半岛、新界等 3 大区域组成。管辖陆地总面积 1104.32 平方公里,截至 2014 年末,总人口约 726.4 万人,人口密度居全球界第三。
</div>
<div class="myBox"><img src="img/B266C.jpg"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,.myTips:after 用于在 myTips 中插入指定的内容(三角形)。在 CSS3 中,after 选择器用于在被选元素的内容后面插入内容,内容通常放在 content 属性中。在此实例中,由于只需要插入部分三角形,因此 content 属性为空白。

此实例的源文件名是 myHtmlB269.html。



图　559-1

560　使用 focus 选择器设置焦点文本框的边框线

此实例主要通过在 focus 选择器中设置 outline-color 属性实现焦点文本框显示红色的边框线以区别普通文本框。当在 Google Chrome 浏览器中显示该页面时,如果"用户名称:"文本框获得焦点,则其边框线显示红色,如图 560-1 所示;如果"联系电话:"文本框获得焦点,则其边框线也将显示红色,如图 560-2 所示;如果"用户密码:"文本框获得焦点,则其边框线仍将显示红色。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <style type="text/css">
  input {width: 200px;}
  input:focus { outline-color: red;}  /*设置焦点文本框的边框线为红色*/
```

```
</style></head>
<body><center><form action="" method="post"><br>
  <label id="myUser">用户名称:<input type="text"></label><br><br>
  <label id="myPassword">用户密码:<input type="text"></label><br><br>
  <label id="myPhone">联系电话:<input type="text"></label>
</form></center></body></html>
```



图 560-1

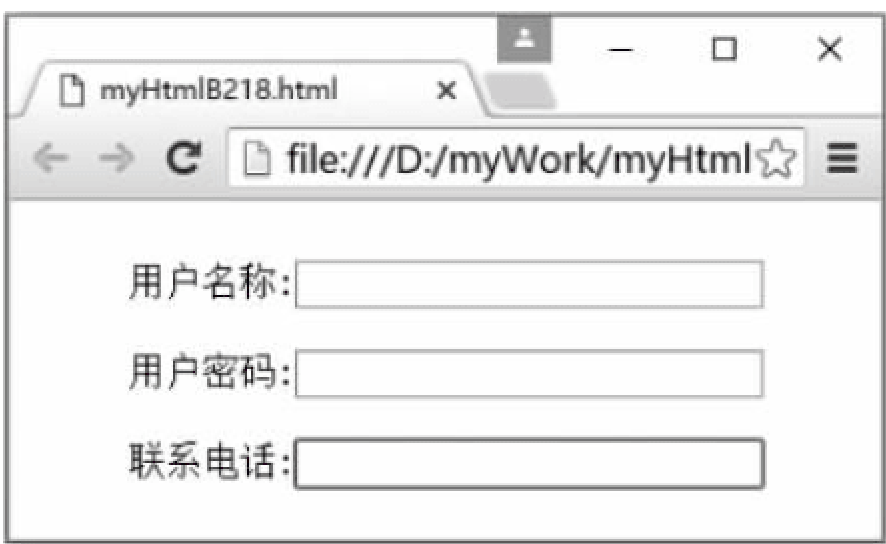


图 560-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，input:focus 选择器用于选取获得焦点的文本框；outline-color:red 用于设置文本框的边框线为红色。在 CSS 中，outline-color 属性用于设置元素轮廓的颜色，该属性支持的属性值的意义如下。

(1) color_name：该属性值规定颜色值为颜色名称的轮廓颜色(例如 red)。

(2) hex_number：该属性值规定颜色值为十六进制值的轮廓颜色(例如#FF0000)。

(3) rgb_number：该属性值规定颜色值为 rgb 代码的轮廓颜色(例如 rgb(255,0,0))。

(4) invert：该属性值是默认值，执行颜色反转(逆向的颜色)，可使轮廓在不同的背景颜色中都可见。

outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时这个宽度才会起作用，如果样式为 none，宽度实际上会重置为 0。outline-width 属性支持的属性值的意义如下。

(1) thin：该属性值规定细轮廓。

(2) medium：该属性值是默认值，规定中等的轮廓。

(3) thick：该属性值规定粗轮廓。

(4) length：该属性值允许规定轮廓粗细值。

outline-style 属性用于设置元素的整个轮廓样式，样式不能是 none，否则轮廓不会出现。outline-style 属性支持的属性值如表 560-1 所示。

表 560-1 outline-style 属性支持的属性值

属性值	意义
none	默认，定义无轮廓
dotted	定义点状的轮廓
dashed	定义虚线轮廓
solid	定义实线轮廓
double	定义双线轮廓，双线的宽度等同于 outline-width 的值
groove	定义 3D 凹槽轮廓，此效果取决于 outline-color 值
ridge	定义 3D 凸槽轮廓，此效果取决于 outline-color 值
inset	定义 3D 凹边轮廓，此效果取决于 outline-color 值
outset	定义 3D 凸边轮廓，此效果取决于 outline-color 值
inherit	规定应该从父元素继承轮廓样式的设置

实际上，outline-color、outline-width、outline-style 这 3 个属性的属性值可以合并到 outline 属性中，例如 outline：red solid medium。

此实例的源文件名是 myHtmlB218.html。

561　使用 selection 选择器突出显示选中文本

此实例主要通过在 selection 选择器中定制阴影文字样式实现以 3D 阴影效果显示选中的文本。当在 Google Chrome 浏览器中显示该页面时，使用鼠标选中的文本"花秋月何"的 3D 阴影显示效果如图 561-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
   p{ font: bold 60px/100 % "微软雅黑", "Lucida Grande", "Lucida Sans", Helvetica, Arial, Sans; }
   /* 设置选中文本的样式 */
   p::selection {text - shadow: - 1px - 1px 0 #FFF, 1px 1px 0 #333, 1px 1px 0 #444;
    background - color: lightgray;}              /* 创建阴影文字 */
   body { background - color: gray; }            /* 设置灰色背景 */
 </style></head>
<body><center><p>春花秋月何时了</p></center></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，selection 选择器用于定义用户的鼠标已选择文本的样式，在此选择器中能够设置的属性相当有限，大多数 CSS 样式属性设置以后都不起作用，但此实例设置的 text-shadow 属性已经测试通过。在 CSS3 中，text-shadow 属性支持给文字加上阴影，从而增加文字的质感而不用使用任何图片，text-shadow 属性的语法格式如下。



图　561-1

```
text - shadow:color length length length
```

其中，color 表示颜色；length 分别按顺序指"X 轴方向长度　Y 轴方向长度　阴影模糊半径"，正值在 X 轴表示向右，负值表示向左，同样的道理，Y 轴负值表示向上。其中任意一个值可以为零也可为空（将做默认处理）。

此实例的源文件名是 myHtmlB239.html。

562　使用 hover 选择器定制选中元素的样式

此实例主要通过使用 hover 选择器实现定制当前图像的选中样式。当在 Google Chrome 浏览器中显示该页面时，如果鼠标指针悬浮在第二幅图像上，则该图像的背后将出现阴影，如图 562-1 所示；如果鼠标指针悬浮在第三幅图像上，则该图像的背后也将出现阴影，如图 562-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  img:hover{box - shadow:5px 5px 2px gray; cursor:pointer;}
```

```
    img{margin:5px;border - radius:10px; width: 100px;height:150px;}
  </style></head>
<body><div><img src = "img/B012A.jpg"/><img src = "img/B012B.jpg"/>
      <img src = "img/B012C.jpg"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，hover 选择器主要用于当页面中的 img 元素处于 hover 状态时设置其背景为阴影并且鼠标指针呈现手指形状，常用的选择器有 E:hover、E:active、E:focus、E:enabled、E:disabled、E:read-only、E:read-write、E:checked、E:selection、E:default、E:indeterminate、E:invalid、E:valid、E:required、E:optional、E:in-range、E:out-of-range。

此实例的源文件名是 myHtmlB022.html。



图 562-1



图 562-2

563 使用 empty 选择器定制元素内容空白时的样式

此实例主要通过使用 empty 选择器实现定制元素内容空白时的样式。当在 Google Chrome 浏览器中显示该页面时，“原产地”空白的两个单元格的背景颜色被设置为天蓝色，如图 563-1 所示。有关此实例的主要代码如下。

myHtmlB010.html

file:///D:/myWork/myHtmlB010

葡萄酒名称	原产地	葡萄品种
罗博克葡萄酒		歌海娜
科洛文红葡萄酒	法国	丹魄
马蒂尔伯爵葡萄酒		朗布罗斯科

图 563-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
  <style>
    table { margin: 0 auto;width: 400px; font - size: 14px;
            border - collapse: collapse;text - align: center;}
    :empty {background - color: lightblue;}
  </style></head>
<body><div style = "width:400px"><table id = "mytable" border = "1" >
    <tr style = "background - color: lightgray; font - size: 16px;">
      <td>葡萄酒名称</td><td>原产地</td><td>葡萄品种</td></tr>
```

```
  <tr><td>罗博克葡萄酒</td><td></td><td>歌海娜</td></tr>
  <tr><td>科洛文红葡萄酒</td><td>法国</td><td>丹魄</td></tr>
  <tr><td>马蒂尔伯爵葡萄酒</td><td></td><td>朗布罗斯科</td></tr></table></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，empty 选择器的主要作用就是当单元格的内容空白时设置其背景为天蓝色。

此实例的源文件名是 myHtmlB010.html。

564　使用 not 选择器自定义子结构元素的样式

此实例主要通过使用 not 选择器实现对某个结构元素使用样式，但是排除这个结构元素(div)下面的子结构元素(p)使用该样式。当在 Google Chrome 浏览器中显示该页面时，(div)唐诗的标题采用大字体、黑色显示，其下的内容(p)使用小字体、绿色显示，如图 564-1 所示。有关此实例的主要代码如下。



图　564-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style>
  div.myFrame { float: left;margin-left: 30px; width: 300px;padding: 10px; border: 1px solid #BFBFBF;
background-color: white; box-shadow: 2px 2px 3px gray;text-align: center; font-size: 24px;font-
weight: bold;} /* 创建有阴影的盒子 */
  body *:not(div) {color: green; font-size: 14px;}
 </style></head>
<body><div class="myFrame">锦瑟
 <p>锦瑟无端五十弦,一弦一柱思华年。</p><p>庄生晓梦迷蝴蝶,望帝春心托杜鹃。</p>
 <p>沧海月明珠有泪,蓝田日暖玉生烟。</p><p>此情可待成追忆,只是当时已惘然。</p> </div></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，not(div)选择器的主要作用是设置 div 下面的 p 元素的字体大小为 14px、字体颜色为绿色。

此实例的源文件名是 myHtmlB011.html。

565　使用 only-child 选择器定制唯一子元素

此实例主要通过使用 only-child 选择器定制在元素中只有一个子元素的样式。当在 Google Chrome 浏览器中显示该页面时，在 3 组榜单中只有一本图书的榜单书名以定制的隶书和浅蓝色背景

显示，其他两个榜单的书名以正常样式显示，如图 565-1 所示。有关此实例的主要代码如下。



图 565-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  p:only-child{background-color: lightblue; border-radius:5px;
   font-weight: bold;font-family: 华文隶书; }
  p{padding-left: 8px;width:340px ; margin:10px ;}
 </style></head>
<body><h3>12 月份新上榜科普图书</h3><div><p>科学的旅程</p><p>无言的宇宙：隐藏在 24 个数学公式背后的故事</p><p>中国国家地理百科全书</p></div>
   <h3>12 月份新上榜计算机图书</h3><div><p>JavaScript 权威指南(第 6 版)</p></div>
   <h3>12 月份新上榜经管图书</h3><div><p>移动互联网思维</p><p>移动互联网时代的新 4C 法则</p>
</div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，p:only-child 选择器用于设置元素中符合要求的唯一子元素以隶书字体和浅蓝色背景显示。

此实例的源文件名是 myHtmlB021.html。

566 使用 first-child 选择器定制开始子元素

此实例主要通过使用 first-child 选择器定制列表中的第一个子元素的样式。当在 Google Chrome 浏览器中显示该页面时，5 本图书的第 1 本图书将以定制的灰色背景显示，如图 566-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  li:first-child{background-color: lightgray; border-radius:5px;}
  li{list-style-type: none;padding-left: 8px;width:340px ;margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
```

```
<ul><li>科学的旅程</li>
    <li>万物解释者：复杂事物的极简说明书</li>
    <li>地球与太空：美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学：315 个烧脑游戏玩通数学史</li>
<li>无言的宇宙：隐藏在 24 个数学公式背后的故事</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:first-child 选择器用于设置列表中第一个列表项目的背景为浅灰色，并且圆角。此前如果需要对列表中的第一个列表项目指定不同的背景色，采用的做法是直接给第一个列表项目设置 class 属性，然后通过 class 属性定制样式。

此实例的源文件名是 myHtmlB013.html。



图　566-1

567　使用 last-child 选择器定制末尾子元素

此实例主要通过使用 last-child 选择器定制列表中的最后一个子元素的样式。当在 Google Chrome 浏览器中显示该页面时，最后一本图书的名称将以定制的灰色背景显示，如图 567-1 所示。有关此实例的主要代码如下。



图　567-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  li:last - child{background - color: lightgray;border - radius:5px;}
```

```
  li{list-style-type: none;padding-left: 8px;width:340px ;margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
<ul><li>科学的旅程</li>
    <li>万物解释者: 复杂事物的极简说明书</li>
    <li>地球与太空: 美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学: 315 个烧脑游戏玩通数学史</li>
    <li>无言的宇宙: 隐藏在 24 个数学公式背后的故事</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:last-child 选择器用于设置列表中最后一个列表项目的背景为浅灰色，并且圆角。此前如果需要对列表中的最后一个列表项目指定不同的背景色，采用的做法是直接给有关的列表项目设置 class 属性，然后通过 class 属性定制样式。

此实例的源文件名是 myHtmlB014.html。

568 使用 nth-child 选择器定制指定序号元素

此实例主要通过使用 nth-child 选择器定制列表中指定序号的子元素的样式。当在 Google Chrome 浏览器中显示该页面时，第二本图书的名称将以定制的灰色背景显示，如图 568-1 所示。有关此实例的主要代码如下。



图 568-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  li:nth-child(2){background-color: lightgray;  border-radius:5px;}
  li{list-style-type: none;padding-left: 8px;
     width:340px;margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
<ul><li>科学的旅程</li>
    <li>万物解释者: 复杂事物的极简说明书</li>
    <li>地球与太空: 美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学: 315 个烧脑游戏玩通数学史</li>
    <li>无言的宇宙: 隐藏在 24 个数学公式背后的故事</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:nth-child(2)选择器用于设置列表中第二个列表项目的背景为浅灰色，并且圆角。注意，li:nth-child(2)的 2 表示按照默认的顺序顺数

第二个子元素，li:nth-last-child(2)的2表示按照默认的顺序倒数第二个子元素，这两个选择器的计数基准完全不同，因此指向的是不同的子元素。

此实例的源文件名是myHtmlB015.html。

569　使用nth-last-child选择器定制倒数元素

此实例主要通过使用nth-last-child选择器定制列表中倒数指定序号的子元素的样式。当在Google Chrome浏览器中显示该页面时，倒数第二本图书的名称将以定制的灰色背景显示，如图569-1所示。有关此实例的主要代码如下。

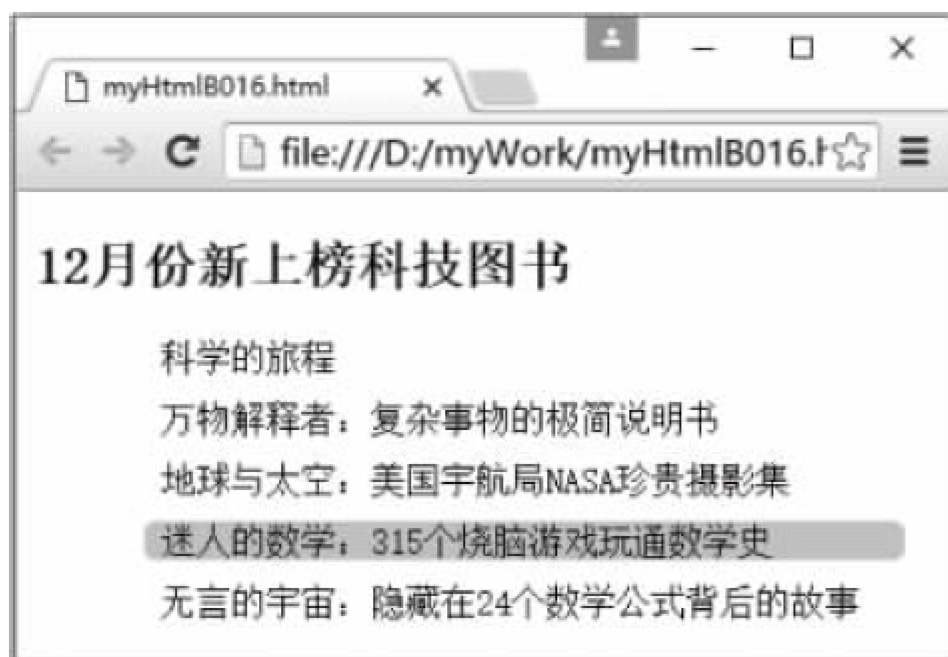


图　569-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  li:nth-last-child(2){ background-color: lightgray;border-radius:5px;}
  li{ list-style-type: none;padding-left: 8px; width:340px;margin:10px ;}
 </style></head>
<body><h2>12月份新上榜科技图书</h2>
<ul><li>科学的旅程</li>
    <li>万物解释者：复杂事物的极简说明书</li>
    <li>地球与太空：美国宇航局NASA珍贵摄影集</li>
    <li>迷人的数学：315个烧脑游戏玩通数学史</li>
    <li>无言的宇宙：隐藏在24个数学公式背后的故事</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:nth-last-child(2)选择器用于设置列表中倒数第二个列表项目的背景为浅灰色，并且圆角。注意，li:nth- child(2)的2表示按照默认顺序的第二个子元素，li:nth-last-child(2)的2表示按照默认的顺序倒数第二个子元素。

此实例的源文件名是myHtmlB016.html。

570　使用target选择器定制目标元素的样式

此实例主要通过使用target选择器定制目标元素的样式。当在Google Chrome浏览器中显示该页面时，单击“黑海夺金”超链接，则下面的“黑海夺金”电影海报图像将旋转17°并显示阴影，如图570-1所示；单击“衰鬼刑警”超链接，则下面的“衰鬼刑警”电影海报图像也将旋转17°并显示阴影，如图570-2所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  :target { transform: rotate(17deg);border:1px solid #BFBFBF;
            box-shadow:2px 2px 3px gray;}
  img {margin: 20px;}
 </style></head>
<body>
<p id="menu" style="margin-left: 200px">
   <a href="#myMovie1">终极硬汉</a> | <a href="#myMovie2">黑海夺金</a> |
   <a href="#myMovie3">衰鬼刑警</a></p>
<div><img id="myMovie1" src="img/B012A.jpg"/>
   <img id="myMovie2" src="img/B012B.jpg"/>
   <img id="myMovie3" src="img/B012C.jpg"/></div></body></html>
```



图 570-1



图 570-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，target 选择器主要用于对页面中的某个 target 元素(该元素的 id 被当作页面中的超链接来使用)指定样式，该样式只有在用户单击了页面中的超链接，并且跳转到 target 元素后起作用。在此实例中，target 选择器的主要功能是设置目标元素的样式为旋转 17°并带有阴影和边框线。

此实例的源文件名是 myHtmlB012.html。

571 使用属性选择器筛选超链接并追加内容

此实例主要通过使用属性选择器(a[href * =jd]:after)实现筛选超链接并在其后追加内容。当在 Google Chrome 浏览器中显示该页面时将对 5 个超链接的 href 属性进行筛选，并在其后添加符合要求的文字说明，如图 571-1 所示。有关此实例的主要代码如下。

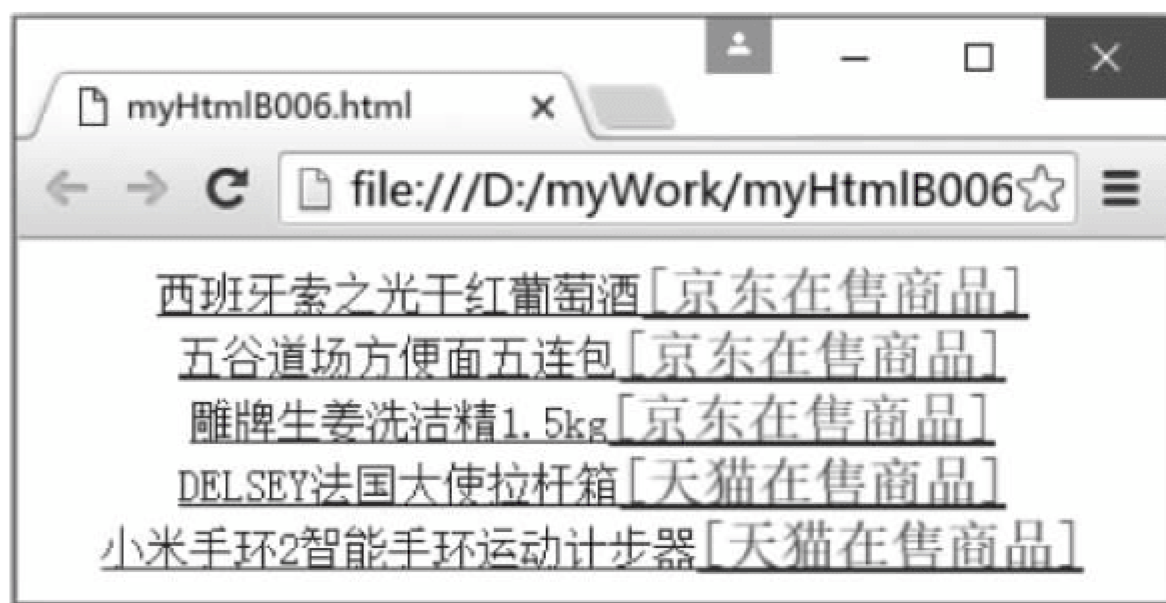


图 571-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a[href * = jd]:after{content:"[京东在售商品]"; color: red; font-size: 20px; }
  a[href * = tmall]:after{content:"[天猫在售商品]";color: green; font-size: 20px; }
 </style></head>
<body><div style="text-align:center;margin-top:5px">
 <a href="http://item.jd.com/2212217.html">西班牙索之光干红葡萄酒</a><br>
 <a href="http://item.jd.com/822935.html">五谷道场方便面五连包</a><br>
 <a href="http://item.jd.com/1521193.html">雕牌生姜洗洁精 1.5kg</a><br>
 <a href="https://detail.tmall.com/item.htm?spm=a220m.1000858.1000725.8.5lfwcH&id=44538838610&skuId=98422660486&areaId=500000&cat_id=55750010&rn=ccaaf2a9cee13ac7e5256dc67874c62b&user_id=1646318507&is_b=1">DELSEY 法国大使拉杆箱</a><br>
 <a href="https://detail.tmall.com/item.htm?spm=a220m.1000858.1000725.13.nu63d4&id=533028473699&skuId=3184920257472&areaId=500000&cat_id=56148012&rn=3293d2aa56dfe47b95ee67d28d50675d&user_id=1807289316&is_b=1">小米手环 2 智能手环运动计步器</a>
<br></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，a[href * =jd]:after 属性选择器表示在超链接中筛选 href 属性中包含 jd 的超链接，然后在其后追加 content，并设置文字颜色和尺寸。CSS3 中还有两个支持通配符的属性选择器，[att ^ =val]属性选择器用于筛选 att 属性值的开始字符是 val 的元素；[att $ =val]属性选择器用于筛选 att 属性值的结尾字符是 val 的元素。

此实例的源文件名是 myHtmlB006.html。

572 使用属性选择器筛选超链接并插入元素

此实例主要通过使用属性选择器(a[href*=jd]:before)实现筛选超链接并在其前面插入图像。当在 Google Chrome 浏览器中显示该页面时将对 5 个超链接的 href 属性进行筛选,如果该超链接表示的是京东在售商品,则在该超链接前面添加京东 logo 图像;如果该超链接表示的是天猫在售商品,则在该超链接前面添加天猫 logo 图像,如图 572-1 所示。有关此实例的主要代码如下。



图 572-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  a[href*=jd]:before {content: url(img/B025A.jpg);} /*插入京东 logo 图像*/
  a[href*=tmall]:before { content: url(img/B025B.jpg);} /*插入天猫 logo 图像*/
 </style></head>
<body><div style="text-align:left;margin-top:5px">
  <a href="http://item.jd.com/2212217.html">西班牙索之光干红葡萄酒</a><br>
  <a href="http://item.jd.com/822935.html">五谷道场方便面五连包</a><br>
  <a href="http://item.jd.com/1521193.html">雕牌生姜洗洁精 1.5kg</a><br>
  <a href="https://detail.tmall.com/item.htm?spm=a220m.1000858.1000725.8.51fwcH&id=44538838610&skuId=98422660486&areaId=500000&cat_id=55750010&rn=ccaaf2a9cee13ac7e5256dc67874c62b&user_id=1646318507&is_b=1">DELSEY 法国大使拉杆箱</a><br>
  <a href="https://detail.tmall.com/item.htm?spm=a220m.1000858.1000725.13.nu63d4&id=533028473699&skuId=3184920257472&areaId=500000&cat_id=56148012&rn=3293d2aa56dfe47b95ee67d28d50675d&user_id=1807289316&is_b=1">小米手环 2 智能手环运动计步器</a><br></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,a[href*=jd]:before 属性选择器表示在超链接中筛选 href 属性中包含 jd 的超链接,然后在其前面插入图像;a[href*=tmall]:before 属性选择器表示在超链接中筛选 href 属性中包含 tmall 的超链接,然后在其前面插入图像。注意,如果 content 属性表示的是纯文本,则应该为纯文本添加双引号,例如 content:"京东在售商品";如果需要在超链接的后面添加图像,则应该使用 a[href*=jd]:after 属性选择器。

此实例的源文件名是 myHtmlB025.html。

573 使用属性选择器筛选超链接并禁止插入

此实例主要在超链接中指定类别 myNone,并使用属性选择器(a[href*=jd].myNone:before),从而实现筛选超链接并禁止在其前面插入图像或文字等内容。当在 Google Chrome 浏览器中显示该

页面时将对5个超链接的href属性进行筛选，如果该超链接表示的是京东在售商品，则在该超链接前面添加京东logo图像，但是如果该超链接的类别为myNone，则禁止添加京东logo图像，例如“五谷道场方便面五连包”；如果该超链接表示的是天猫在售商品，则在该超链接前面添加天猫logo图像，如图573-1所示。有关此实例的主要代码如下。

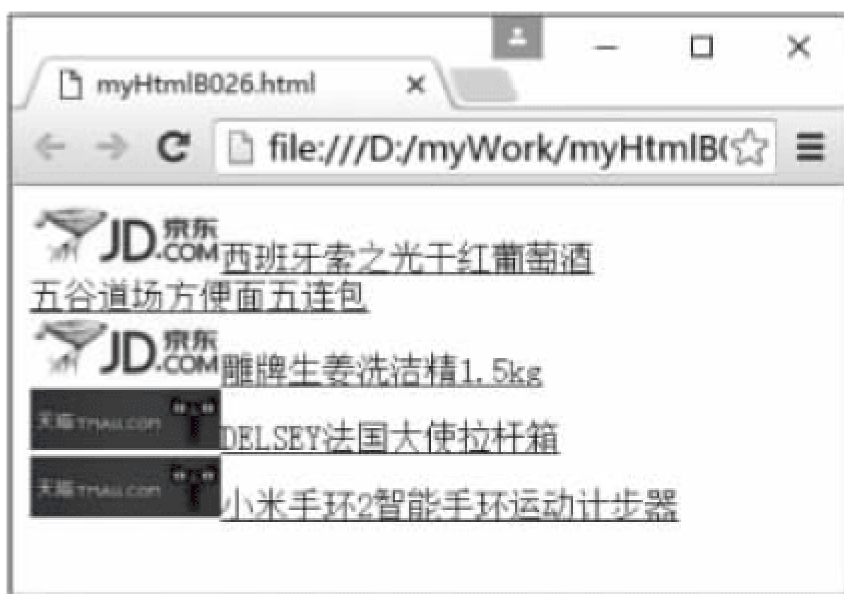


图　573-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  a[href * = jd]:before {content: url(img/B025A.jpg); }       /* 插入京东 logo 图像 */
  a[href * = jd].myNone:before { content:normal; }           /* 禁止插入 logo 图像 */
  a[href * = tmall]:before { content: url(img/B025B.jpg);}   /* 插入天猫 logo 图像 */
 </style></head>
<body><div style = "text - align:left;margin - top:5px">
 <a href = "http://item.jd.com/2212217.html">西班牙索之光干红葡萄酒</a><br>
 <a href = "http://item.jd.com/822935.html" class = "myNone">五谷道场方便面五连包</a><br>
 <a href = "http://item.jd.com/1521193.html">雕牌生姜洗洁精 1.5kg</a><br>
 <a href = "https://detail.tmall.com/item.htm?spm = a220m.1000858.1000725.8.51fwcH&id = 44538838610&s
kuId = 98422660486&areaId = 500000&cat_id = 55750010&rn = ccaaf2a9cee13ac7e5256dc67874c62b&user_id =
1646318507&is_b = 1"> DELSEY 法国大使拉杆箱</a><br>
 <a href = "https://detail.tmall.com/item.htm?spm = a220m.1000858.1000725.13.nu63d4&id = 53302847369
9&skuId = 3184920257472&areaId = 500000&cat_id = 56148012&rn = 3293d2aa56dfe47b95ee67d28d50675d&user_
id = 1807289316&is_b = 1">小米手环 2 智能手环运动计步器</a><br></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，a[href * =jd].myNone:before属性选择器表示在超链接中筛选href属性中包含jd的超链接，并禁止在其前面插入图像。

此实例的源文件名是myHtmlB026.html。

574　使用属性选择器筛选数据实现列表过滤

此实例主要通过使用CSS3的属性选择器实现筛选数据并选择城市。当在Google Chrome浏览器中显示该页面时，单击“所在城市：”文本框则会滑出一个下拉列表框，可以直接在下拉列表框中选择城市；也可以在文本框中输入拼音或汉字，例如“ou”，则将在下拉列表中显示拼音包含“ou”的所有城市，选择符合要求的城市，该城市就会自动填充到文本框中，如图574-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .search { width: 180px; padding: 5px; - webkit - box - sizing: content - box;}
```

```
    .datalist {display: block;visibility: hidden; width: 193px;
              background-color: #FFF; overflow: hidden;
              box-shadow: 0 1px #CCC, 1px 0 #CCC, -1px 0 #CCC, 0 -1px #CCC;}
    .search:focus + .datalist {visibility: visible;}
    .list { margin-top: -1px; padding: 4px 10px; border-top: 1px solid #EEE;}
    p { float: left; text-align: center;line-height: 2px;}
   </style></head>
 <body><p>所在城市: <div><input type="search" class="search" id="city" placeholder=
 "输入省会或直辖市名称"/>
    <label class="datalist" for="city">
     <div class="list" data-index="重庆市 chongqing">重庆市</div>
     <div class="list" data-index="哈尔滨市 haerbing">哈尔滨市</div>
     <div class="list" data-index="长春市 changchun">长春市</div>
     <div class="list" data-index="兰州市 lanzhou">兰州市</div>
     <div class="list" data-index="北京市 beijing">北京市</div>
     <div class="list" data-index="杭州市 hangzhou">杭州市</div>
     <div class="list" data-index="长沙市 changsha">长沙市</div>
     <div class="list" data-index="沈阳市 shenyang">沈阳市</div>
     <div class="list" data-index="成都市 chengdu">成都市</div>
     <div class="list" data-index="合肥市 hefei">合肥市</div>
     <div class="list" data-index="天津市 tianjin">天津市</div>
     <div class="list" data-index="西安市 xian">西安市</div>
     <div class="list" data-index="武汉市 wuhan">武汉市</div>
     <div class="list" data-index="济南市 jinan">济南市</div>
     <div class="list" data-index="广州市 guangzhou">广州市</div>
     <div class="list" data-index="南京市 nanjing">南京市</div>
     <div class="list" data-index="上海市 shanghai">上海市</div>
     <div class="list" data-index="昆明市 kunming">昆明市</div>
     <div class="list" data-index="郑州市 zhengzhou">郑州市</div>
     <div class="list" data-index="贵阳市 guiyang">贵阳市</div>
     <div class="list" data-index="西宁市 xining">西宁市</div>
     <div class="list" data-index="海口市 haikou">海口市</div>
     <div class="list" data-index="南昌市 nanchang">南昌市</div>
     <div class="list" data-index="香港 特区 xianggang">香港 特区</div>
     <div class="list" data-index="澳门 特区 aomen">澳门 特区</div></label>
   <script language="javascript">
    if (document.addEventListener) {
     var eleStyle = document.createElement("style"),
         eleInput = document.querySelector("#city"),
         eleDatalist = document.querySelector(".datalist");
     document.querySelector("head").appendChild(eleStyle);
     eleInput.addEventListener("input", function() {              //文本框输入
      var val = this.value.trim().toLowerCase();
      if (val !== '') { eleStyle.innerHTML = '.list:not([data-index *= "' + this.value + '"]) { display: none; }';
      } else { eleStyle.innerHTML = ''; } });
    eleDatalist.addEventListener("mousedown", function(event) {     //单击确定
     eleInput.value = event.target.innerHTML;
     eleInput.blur();  });}
   </script></div></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,list:not([data-index *="ou"]){ display: none; }表示隐藏在 data-index 中不含有 ou 的所有城市。data-index *="ou"的本意是选择自定义属性 data-index 中含有 ou 的所有城市,但它前面有个 not 过滤器,因此取反。CSS3 常用的

属性选择器有[att](有该属性)、[att=xxx](属性值是 xxx)、[att ^=xxx](属性值是以 xxx 开头)、[att $ =xxx](属性值是以 xxx 结尾)、[att * =xxx](属性值包含 xxx)。

此实例的源文件名是 myHtmlB162.html。

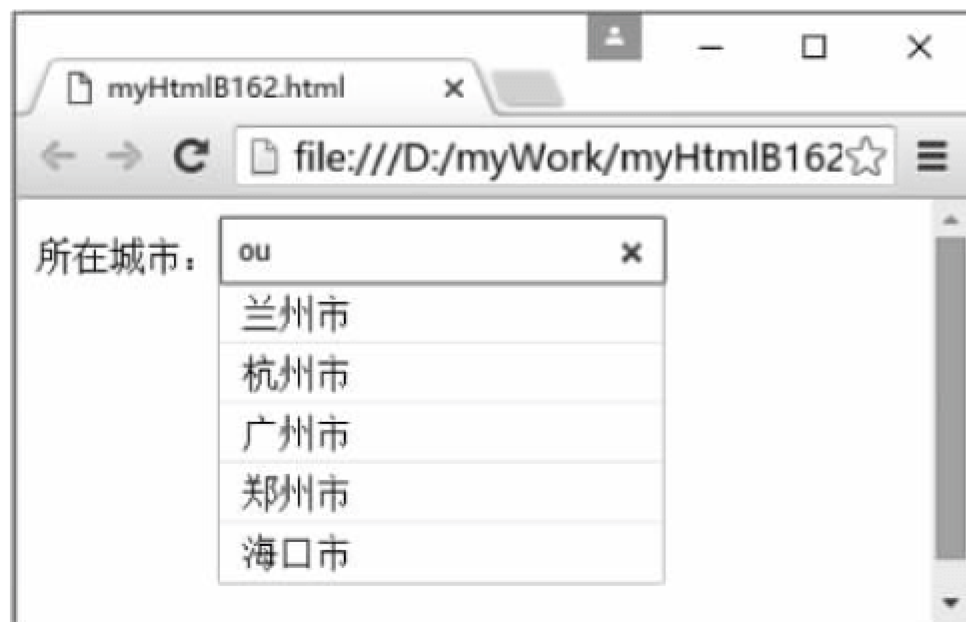


图　574-1

575　使用兄弟选择器定制同级指定元素的样式

此实例主要通过使用 div ～ p 兄弟选择器实现定制同级指定元素的样式。当在 Google Chrome 浏览器中显示该页面时,《移动互联网思维》和《移动互联网时代的新 4C 法则》与《新上榜科普图书》div 元素是同级兄弟关系,前两本书名是 p 元素,因此这两本书的书名以定制的隶书粗体字显示;《科学的旅程》《无言的宇宙:隐藏在 24 个数学公式背后的故事》《中国国家地理百科全书》《JavaScript 权威指南(第 6 版)》虽是 p 元素,但是与《新上榜科普图书》div 元素不是同级兄弟关系,因此不以定制的样式显示;《Python 程序设计入门到实战》虽与《新上榜科普图书》div 元素是同级兄弟关系,但它是 output 元素,而不是 p 元素,因此也不以定制的样式显示,如图 575-1 所示。有关此实例的主要代码如下。



图　575-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  div,p,h3,output{background-color: lightgray;}
```

```
  div ~ p { border - radius:5px; font - weight: bold; font - family: 华文隶书;}
 </style></head>
<body><div><h3>新上榜科普图书</h3>
 <p>科学的旅程</p><p>无言的宇宙: 隐藏在 24 个数学公式背后的故事</p>
 <p>中国国家地理百科全书</p></div>
<div><p>JavaScript 权威指南(第 6 版)</p></div>
<p>移动互联网思维</p><output>Python 程序设计入门到实战</output>
<p>移动互联网时代的新 4C 法则</p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,div ~ p 兄弟选择器用于定制样式以圆角、粗体字和隶书显示元素。在 CSS3 中,兄弟选择器用来指定位于同一个元素之中的某个元素之后的所有其他某个类型的兄弟元素所使用的样式,它的定义格式如下。

```
<子元素>~ <子元素之后的同级兄弟元素>{
//指定样式
}
```

此实例的源文件名是 myHtmlB024.html。

576　使用选择器定制元素的奇数子元素的样式

此实例主要通过使用 nth-child(odd)选择器和 nth-last-child(odd)选择器实现定制列表中顺数为奇数的子元素和倒数为奇数的子元素的样式。当在 Google Chrome 浏览器中显示该页面时,在 6 本图书名称中顺数为奇数的图书书名以灰色背景显示,倒数为奇数的图书书名以浅绿色背景显示并且字体为粗体,如图 576-1 所示。有关此实例的主要代码如下。

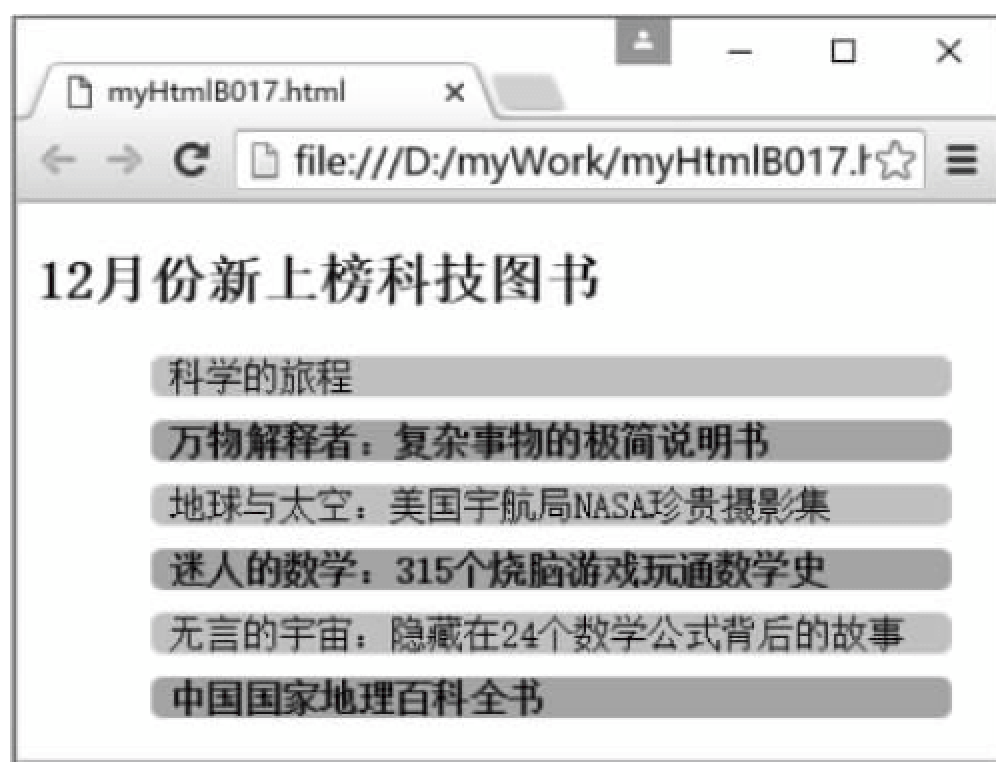


图　576-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  li:nth - child(odd){ background - color: lightgray; border - radius:5px;}
  li:nth - last - child(odd){background - color: lightgreen;
         border - radius:5px;font - weight: bold;}
  li{list - style - type: none;padding - left: 8px;width:340px ;margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
<ul><li>科学的旅程</li>
```

```
    <li>万物解释者：复杂事物的极简说明书</li>
    <li>地球与太空：美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学：315 个烧脑游戏玩通数学史</li>
    <li>无言的宇宙：隐藏在 24 个数学公式背后的故事</li>
    <li>中国国家地理百科全书</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:nth-child(odd)选择器用于设置列表中顺数为奇数的所有列表项目的背景为浅灰色，并且圆角；li:nth-last- child(odd)选择器用于设置列表中倒数为奇数的所有列表项目的背景为浅绿色，字体为粗体并且圆角。

此实例的源文件名是 myHtmlB017.html。

577　使用选择器定制元素的偶数子元素的样式

此实例主要通过使用 nth-child(even)选择器和 nth-last-child(even)选择器实现定制列表中顺数为偶数的子元素和倒数为偶数的子元素的样式。当在 Google Chrome 浏览器中显示该页面时，在 6 本图书名称中，顺数为偶数的图书书名以灰色背景显示，倒数为偶数的图书书名以浅绿色背景显示并且字体为粗体，如图 577-1 所示。有关此实例的主要代码如下。



图　577-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  li:nth-child(even){background-color: lightgray;border-radius:5px;}
  li:nth-last-child(even){background-color: lightgreen;
                          border-radius:5px;font-weight: bold;}
  li{list-style-type: none;padding-left: 8px;width:340px ;margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
<ul><li>科学的旅程</li>
    <li>万物解释者：复杂事物的极简说明书</li>
    <li>地球与太空：美国宇航局 NASA 珍贵摄影集</li>
    <li>迷人的数学：315 个烧脑游戏玩通数学史</li>
    <li>无言的宇宙：隐藏在 24 个数学公式背后的故事</li>
    <li>中国国家地理百科全书</li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，li:nth-child(even)选择器用于设置

列表中顺数为偶数的所有列表项目的背景为浅灰色，并且圆角；li:nth-last- child(even)选择器用于设置列表中倒数为偶数的所有列表项目的背景为浅绿色，字体为粗体并且圆角。在 CSS3 中，nth-child(n)选择器匹配父元素中的第 n 个子元素，n 可以是一个数字，一个关键字(例如 odd 、even 等)，或者一个公式(例如 3n+1)。

此实例的源文件名是 myHtmlB018.html。

578 使用选择器定制元素的倍数子元素的样式

此实例主要通过使用 nth-of-type(3n+1)选择器和 nth-last-of-type(3n+1)选择器实现定制元素中顺数为(3n+1)的子元素和倒数为(3n+1)的子元素的样式。当在 Google Chrome 浏览器中显示该页面时，在 6 本图书名称中，顺数为(3n+1)的图书分别是 1、4，因此这两个图书书名以灰色背景显示，倒数为(3n+1)的图书分别是 6、3，因此这两个图书书名以浅蓝色背景显示并且字体为隶书，如图 578-1 所示。有关此实例的主要代码如下。



图 578-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*顺数倍数(3n+1)选择器*/
  p:nth-of-type(3n+1){ background-color: lightgray;border-radius:5px;}
  /*倒数倍数(3n+1)选择器*/
  p:nth-last-of-type(3n+1){ background-color: lightblue; border-radius:5px;
font-weight: bold; font-family: 华文隶书; }
  p{padding-left: 8px;width:340px ; margin:10px ;}
 </style></head>
<body><h2>12 月份新上榜科技图书</h2>
<div><p>科学的旅程</p>
    <p>万物解释者: 复杂事物的极简说明书</p>
    <p>地球与太空: 美国宇航局 NASA 珍贵摄影集</p>
    <p>迷人的数学: 315 个烧脑游戏玩通数学史</p>
    <p>无言的宇宙: 隐藏在 24 个数学公式背后的故事</p>
    <p>中国国家地理百科全书</p></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，p:nth-of-type(3n+1)选择器用于设置元素(div)中顺数为(3n+1)的所有子元素(p)的背景为浅灰色，并且圆角；p:nth-last-of-type(3n+1)选择器用于设置元素(div)中倒数为(3n+1)的所有子元素(p)的背景为浅蓝色，字体为隶书并且圆角。在 CSS3 中，nth-of-type(n)选择器用于匹配同类型中的第 n 个同级兄弟元素，n 可以是一个数

字，一个关键字（例如 odd 、even 等），或者一个公式（例如 3n＋1）。

此实例的源文件名是 myHtmlB019.html。

579 使用选择器实现表格隔行错色显示

此实例主要通过使用 nth-of-type(2n＋0)选择器和 nth-child(2n＋1)选择器实现动态生成的任意行数的表格隔行错色显示。当在 Google Chrome 浏览器中显示该页面时，表格的奇数行以灰色背景显示，偶数行以浅绿色背景、隶书字体显示，如图 579-1 所示。有关此实例的主要代码如下。

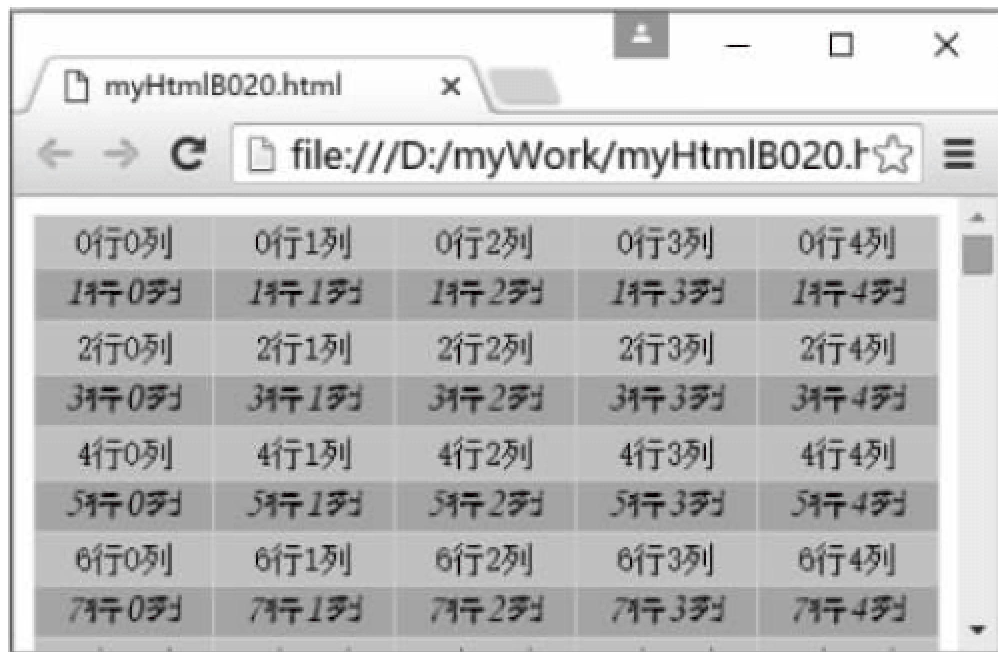


图 579-1

```
<!doctype html><html><head><meta charset = "UTF-8">
 <script src = "js/jquery-3.1.1.min.js"></script><script language = "javascript">
  $(function() {
   var myTable = document.createElement("table");
   for (var i = 0; i < 500; i++) {                    //500 行
    var myTr = document.createElement("tr");
    for (var j = 0; j < 5; j++) {                     //5 列
     var myTd = document.createElement("td");
     var myText = document.createTextNode(i + "行" + j + "列");
     myTd.style.border = "1px solid #dfdfdf";
     myTd.style.width = "80px";
     myTd.style.height = "20px";
     myTd.style.textAlign = "center";
     myTd.appendChild(myText);
     myTr.appendChild(myTd);                          //向行中添加列(单元格)
    }
    myTable.appendChild(myTr);                        //向表中添加行
   }
   $("#myDiv").append(myTable);                      //在指定位置添加表
  })
 </script>
 <style>
   table { margin: 0 auto;font-size: 14px;
           border-collapse: collapse;text-align: center;}
   tr:nth-of-type(2n+0) { background-color: lightgreen;
     font-family: 华文隶书; font-size: 18px; }         /* 偶数行样式选择器 */
   tr:nth-child(2n+1) { background-color: lightgray; }  /* 奇数行样式选择器 */
 </style></head>
<body><div id = "myDiv"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，tr:nth-of-type(2n+0)选择器用于设置表格中行号为偶数的所有行的背景为浅绿色，字体为隶书；tr:nth- child(2n+1)选择器用于设置表格中行号为奇数的所有行的背景为浅灰色。通过修改公式(例如 2n+1)可以实现表格行以其他样式显示。

此实例的源文件名是 myHtmlB020.html。

580 使用选择器实现下拉列表框的选项隔行错色显示

此实例主要通过使用 nth-child 选择器设置符合条件的选项颜色，从而实现下拉列表框的奇数行选项和偶数行选项分别使用不同的颜色显示。当在 Google Chrome 浏览器中显示该页面时，下拉列表框的奇数行选项显示青色，偶数行选项显示粉色，如图 580-1 所示。有关此实例的主要代码如下。

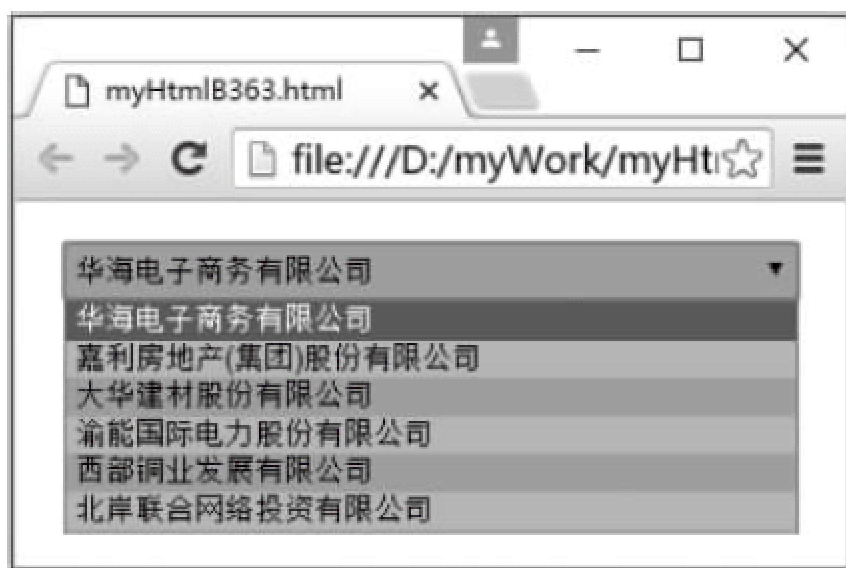


图 580-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  select{width:320px;height:25px;background-color: cyan; margin: 10px; }
  /*设置偶数行选项的背景颜色*/
  select option:nth-child(2n){ background-color: pink;}
 </style></head>
<body><div align="center">
 <select><option  selected>华海电子商务有限公司</option>
        <option>嘉利房地产(集团)股份有限公司</option>
        <option>大华建材股份有限公司</option>
        <option>渝能国际电力股份有限公司</option>
        <option>西部铜业发展有限公司</option>
       <option>北岸联合网络投资有限公司</option></select></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，select option:nth- child(2n) { background-color: pink; }表示设置 select 下拉列表框的所有偶数行选项的背景颜色为粉色。在 CSS3 中，nth-child(n)选择器匹配属于其父元素的第 n 个子元素，不论元素的类型；n 可以是数字、关键字或公式；odd 和 even 是可用于匹配下标是奇数或偶数的子元素的关键字。因此，上面的代码也可以修改成：select option:nth-child(even) {background-color: pink;}。

此实例的源文件名是 myHtmlB363.html。

581 使用选择器定制超范围文本框的显示样式

此实例主要通过使用 input[type="number"]:in-range 和 input[type="number"]: out-of-range 选择器实现定制数字文本框的值在超出限定范围时的外观样式。当在 Google Chrome 浏览器

中显示该页面时，如果在“出生年份(1990—1999):”文本框中输入的值大于1999或者小于1990，则该文本框的背景将出现红色，如图581-1所示；如果在“出生年份(1990—1999):”文本框中输入的值在1990—1999的有效范围内，则该文本框的背景将以正常的白色显示。有关此实例的主要代码如下。

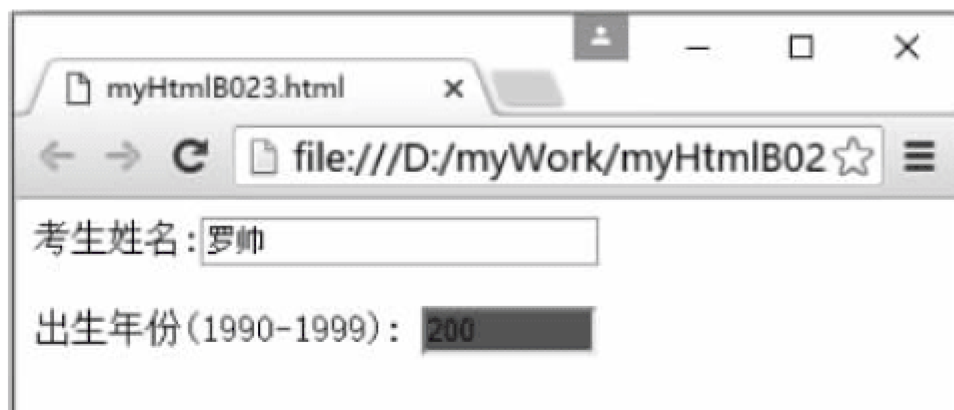


图　581-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  /* 在范围内时背景为白色 */
  input[type = "number"]:in - range{ background - color: white;}
  /* 超出范围内时背景为红色 */
  input[type = "number"]:out - of - range{ background - color: red;}
 </style></head>
<body><form>考生姓名:<input type = "text" name = "myName"/><br><br>
  出生年份(1990 - 1999): <input type = number min = 1990 max = 1999 ><br><br></form>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，input[type="number"]: in-range选择器用于设置数字文本框的值在有效范围内时以白色背景显示；input[type="number"]:out-of-range选择器用于设置数字文本框的值在超出范围时以红色背景显示。

此实例的源文件名是myHtmlB023.html。

582　使用选择器实现内圆弧化的渐变曲线图形

此实例主要通过使用before和after选择器实现内圆弧化的渐变曲线图形。当在Google Chrome浏览器中显示该页面时，内圆弧化的渐变曲线图形效果如图582-1所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style type = "text/css">
  .myBox { margin - top: 50px; z - index: 3; overflow: hidden; background: linear - gradient ( white, darkgreen); width:400px; height:250px; padding - bottom: 25px; }/* 绘制渐变的矩形 */
  .myBox:before{ content:""; display: block; position:relative; top: - 95px; border - radius: 50% ; z - index: 4; width:400px;  height:190px; background - color:white; }/* 绘制上面的椭圆 */
  .myBox:after{ content:""; display: block; position:relative; top: - 64px; border - radius: 50% ; z - index:5; width:400px; height:190px; background - color: white;box - shadow: 2px 2px 8px black inset; }/* 绘制下面的椭圆 */
  /* 设置文字块的基本样式 */
  .myBox div{ position:relative; bottom:80px;font - size:50px; font - weight: bold; }
 </style></head>
<body>
<div align = "center">
    <div class = "myBox" ><div>炫酷实例集锦</div></div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myBox:before{}选择器用于绘制上面的椭圆，myBox:after{}选择器用于绘制下面的椭圆，由于此实例的椭圆没有边框线，并且背景色与容器的颜色相同，因此造成内圆弧的视觉效果，但它们的实际构造如图 582-2 所示，椭圆的其余部分被 overflow: hidden 了。在 CSS3 中，before 选择器用于在被选元素的内容前面插入内容，插入内容在 content 属性中指定；after 选择器用于在被选元素的内容后面插入内容，插入内容在 content 属性中指定。

此实例的源文件名是 myHtmlB346.html。



图 582-1

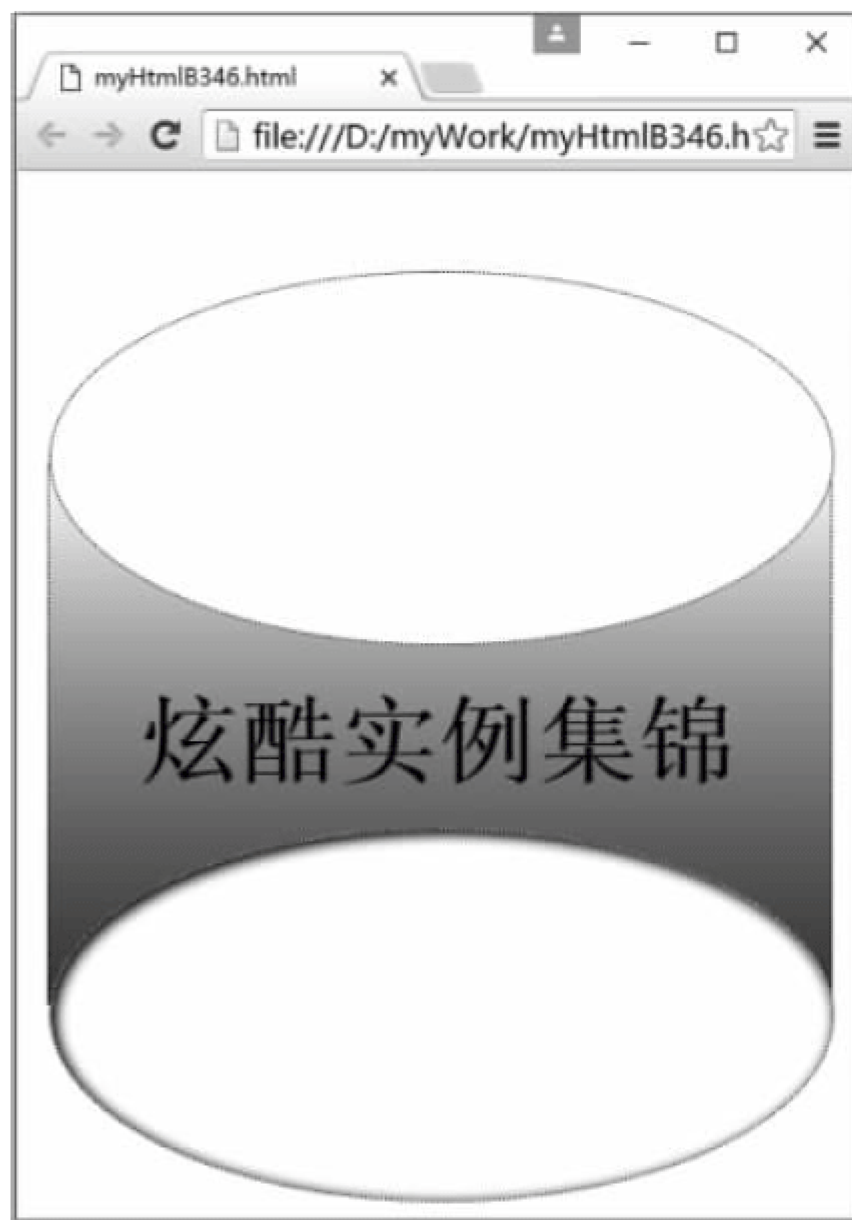


图 582-2

583 使用选择器绘制扇形样式的多级彩虹

此实例主要通过使用 before 和 after 选择器实现绘制扇形样式的多级彩虹。当在 Google Chrome 浏览器中显示该页面时，扇形样式的多级彩虹效果如图 583-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = "UTF-8">
 <style type = "text/css">
  /*设置盒子的基本样式*/
  .myBox { margin: 10px auto; overflow: hidden; width: 400px; height: 200px;  }
  /*绘制半圆形的彩虹*/
  .myRainbow { margin-top: 10px; border-radius: 50%; width: 400px; height: 400px; background: -webkit-radial-gradient(#FFFFFF 80px, #FF6633 100px, #FFFF00 120px, green 140px, #008AFF 160px, purple 180px, cyan 200px); }
  /*以白色切掉半圆形彩虹的左边小部分*/
  .myRainbow:before { content: "";  display: block; width: 0; height: 0; position: relative; border: 200px solid transparent; border-left: 200px solid white; }
  /*以白色切掉半圆形彩虹的右边小部分*/
```

```
  .myRainbow:after { content: ""; display: block;width: 0; height: 0; border: 200px solid transparent;
border-right: 200px solid white; position: relative; top: -400px;}
 </style></head>
<body><div class="myBox"><div class="myRainbow"></div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background：-webkit- radial-gradient(#FFFFFF 80px，#FF6633 100px，#FFFF00 120px，green 140px，#008AFF 160px，purple 180px，cyan 200px)用于绘制多级彩虹，因为盒子的高度等于半径200px，因此显示半圆形的多级彩虹；myRainbow:before{}选择器用于裁剪半圆形彩虹的左边部分，它的实际形状如图583-2的灰色部分所示，在此实例中由于设置其颜色为与盒子背景相同的白色，用白色遮盖半圆形彩虹的左边部分；即在视觉上产生被裁剪的效果。myRainbow：after{}选择器用于裁剪半圆形彩虹的右边部分，它的裁剪原理与myRainbow:before{}选择器的原理相同，仅方向相反。在CSS3中，before选择器用于在被选元素的内容前面插入内容，插入内容在content属性中指定；after选择器用于在被选元素的内容后面插入内容，插入内容在content属性中指定。在此实例中仅使用了两个选择器的形状，并没有使用其content属性。

此实例的源文件名是myHtmlB349.html。



图　583-1



图　583-2

584　使用选择器将按钮拆分成左、右两部分

此实例主要使用before选择器添加内容并设置样式，从而实现将一个按钮拆分成左、右两部分。当在Google Chrome浏览器中显示该页面时，每个按钮将被分成左、右两部分，左半部分显示排名，右半部分显示书名，如图584-1所示；如果单击第2个按钮，则该按钮的左、右两部分将同时下陷，如图584-2所示，然后弹出一个消息框。单击其他按钮将显示类似的效果。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  /*设置盒子的基本样式*/
  .myBox { margin: 15px auto 0px auto; width: 350px; }
  /*设置按钮右半部分的样式*/
  .myBtn { margin: 20px;background-color:#3BB3E0; position:relative; font-size:12px; font-family:
'Open Sans', sans-serif; text-decoration:none; color:#FFF; padding:10px 10px;border: none; border-
```

```
left:solid 1px #2AB7EC; margin-left:55px; background-image: -webkit-linear-gradient(bottom, rgb(44,160,202) 0%, rgb(62,184,229) 100%); -webkit-border-radius: 0 5px 5px 0; -webkit-box-shadow: inset 0px 1px 0px #2AB7EC, 0px 5px 0px 0px #156785, 0px 10px 5px #999; text-align: left;}
  /*设置按钮左半部分的样式*/
  .myBtn::before { content:""; width:35px; height:100%;position:absolute; display: block; padding-top:8px; top:0px; left: -36px; font-size:16px; font-weight:bold; color:#8FD1EA; text-shadow:1px 1px 0px #07526E; border-right:solid 1px #07526E; background-image: -webkit-linear-gradient(bottom, rgb(10,94,125) 0%, rgb(14,139, 184) 100%);-webkit-border-radius: 5px 0 0 5px; -webkit-box-shadow: inset 0px 1px 0px #2AB7EC, 0px 5px 0px 0px #032B3A, 0px 10px 5px #999; text-align: center; }
  /*设置按钮左半部分的内容*/
  #a1.myBtn::before { content:"1"; }
  #a2.myBtn::before { content:"2"; }
  #a3.myBtn::before { content:"3"; }
  #a4.myBtn::before { content:"4"; }
  /*设置单击时的按钮样式*/
  .myBtn:active { top:3px; -webkit-box-shadow: inset 0px 1px 0px #2AB7EC, 0px 2px 0px 0px #156785, 0px 5px 3px #999; }
 </style></head>
<body><div class="myBox" align="center">
  <div align="center" class="myBtn" id="a1" onclick="alert('')">Tomcat权威指南</div>
  <div align="center" class="myBtn" id="a2" onclick="alert('')">MySQL数据库应用从入门到精通</div>
  <div align="center" class="myBtn" id="a3" onclick="alert('')">Oracle Database 12c完全参考手册</div>
  <div align="center" class="myBtn" id="a4" onclick="alert('')">数理统计与数据分析</div></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，myBtn::before{ }选择器用于在按钮的前面插入内容。在此实例中，由于每个按钮的内容均不相同，因此可以将 myBtn::before { }选择器的代码分成两部分，内容部分单独定制，例如#a1.myBtn::before { content:"1"; }。

此实例的源文件名是 myHtmlB371.html。



图 584-1



图 584-2

第8部分 存储

585 使用 localStorage 读取或保存本地数据

此实例主要通过使用 localStorage 实现在本地硬盘中读取或保存页面上的数据。当在浏览器中显示该页面时，在“用户名称:”和“用户密码:”文本框中分别输入内容，单击“在本地保存数据”按钮，则在两个文本框中的内容将被保存到本地硬盘中；重新刷新页面，“用户名称:”和“用户密码:”文本框呈现空白，单击“在本地读取数据”按钮，则将自动在“用户名称:”和“用户密码:”文本框中显示此前保存的内容，如图 585-1 所示。有关此实例的主要代码如下。

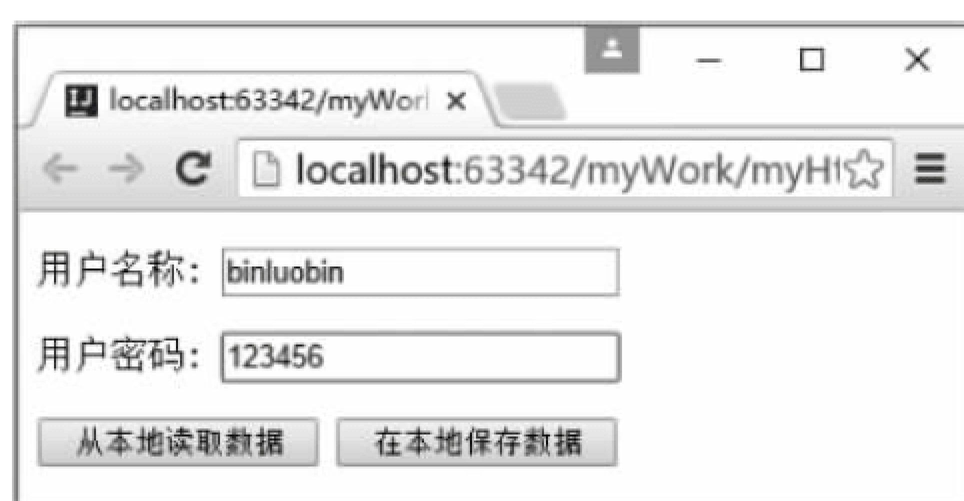


图 585-1

```
<!doctype html><html><head><meta charset = UTF - 8>
 <script src = "js/jquery - 3.1.1.min.js"></script><script language = "javascript">
   $(document).ready(function() {
    $("#myBtnRead").click(function() {          //从本地读取数据
     $("#myUsername").val(localStorage.myUsername);
     $("#myPassword").val(localStorage.myPassword);
    });
    $("#myBtnSave").click(function() {          //在本地保存数据
     var myUsername = $("#myUsername").val();
     localStorage.myUsername = myUsername;
     var myPassword = $("#myPassword").val();
     localStorage.myPassword = myPassword;  });});
 </script></head>
<body><p><label>用户名称:<input id = "myUsername" type = "text"  maxlength = "20">
    </label></p>
 <p><label>用户密码:<input id = "myPassword" type = "text"  maxlength = "20">
```

```
    </label></p>
  <p><input type="button" value="从本地读取数据" id="myBtnRead" style="width: 122px"><input type
="button" value="在本地保存数据" id="myBtnSave" style="width: 122px"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，localStorage 用于将页面上的指定数据保存到客户端本地的硬件（通常是硬盘，也可以是其他硬件设备）设备中，即使浏览器被关闭了，该数据仍然存在，下次打开浏览器访问页面时仍然可以继续使用该数据。sessionStorage 具有与 localStorage 类似的功能，sessionStorage 是将数据保存到 session 对象中。所谓 session，是指用户在浏览某个网站时从进入网站到浏览器关闭所经过的这段时间，也就是用户浏览这个网站所花费的时间。session 对象可以用来保存在这段时间内所要求保存的任何数据。

此实例的源文件名是 myHtmlA074.html。

586 使用 localStorage 修改和保存表格数据

此实例主要通过使用 localStorage 实现在本地硬盘中读取和保存表格 table 中的修改数据。当在浏览器中显示该页面时，单击“从本地读取表格数据”按钮，将从本地硬盘中读取数据并显示在表格中，如图 586-1 所示；任意修改表格的数据，例如将“海马刀红酒开瓶器艾拉提诺”修改为“海马刀红酒”，单击“在本地保存表格数据”按钮，则修改结果将被保存到本地硬盘中，如图 586-2 所示；单击“清空本地数据”按钮，则会删除保存在本地硬盘中的数据修改结果。有关此实例的主要代码如下。

图 586-1

图 586-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  //数据源,JSON 格式
  var myWarehouse = [{"id": 6, "Merchandise": "土老憨香辣豆豉", "Price": "35"},{"id": 7,
"Merchandise": "百草味牛肉干五香牛肉条", "Price": "20"},{"id": 8, "Merchandise": "五谷道场方便面骨汤
蔬菜豉汁辣排面 ", "Price": "15"},{"id": 9, "Merchandise": "法国原瓶限量版红酒巴士底城堡 AOC 级干红葡
萄酒", "Price": "15205"},{"id": 10, "Merchandise": "海马刀红酒开瓶器艾拉提诺", "Price": "35"},{"id":
11, "Merchandise": "果珍阳光甜橙袋装", "Price": "20"},{"id": 12, "Merchandise": "七匹狼(SEPTWOLVES)男
士皮带", "Price": "220"},{"id": 13, "Merchandise": "老榨坊四川风味菜籽油", "Price": "125"}];
   $(document).ready(function() {
    $("#myBtnRead").click(function() {          //从本地读取表格数据
```

```
    var myText = "" + localStorage.myText;
    if(myText.length>20){                      //读取保存在本地硬盘中的数据
     var myParent = myText.split(" + ");
     var tab = "<table id='myTable'>";
     for (i = 0; i < myParent.length - 1; i++) {
      var myElement = myParent[i].split(",");
      tab += "<tr align='center'>"
      for(j = 0;j < myElement.length;j++){
       tab += "<td>" + myElement[j] + "</td>"
      }
      tab += "</tr>";
     }
     tab += "</table>";
     $("#div").html(tab);
     $('#myTable').prop("contenteditable", true);
    }else{//首次访问网页时采用示例数据
     var tab = "<table id='myTable'>";
     $.each(myWarehouse, function (id, item) {//遍历解析 JSON
      tab += "<tr align='center'><td>" + item.id + "</td><td>" + item.Merchandise + "</td><td>" +
item.Price + "</td></tr>"; })
     tab += "</table>";
     $("#div").html(tab);
     $('#myTable').prop("contenteditable", true);
    } });
    $("#myBtnSave").click(function() {     //在本地保存表格数据
     var myText = "";
     for (row = 0; row < $("#myTable").prop('rows').length; row++) {
       myText += "";
      var myCell = $("#myTable").find("tr:eq(" + row + ")").find("td:eq(0)").text();
      myText += myCell + ",";
      var myCell = $("#myTable").find("tr:eq(" + row + ")").find("td:eq(1)").text();
       myText += myCell + ",";
       var myCell = $("#myTable").find("tr:eq(" + row + ")").find("td:eq(2)").text();
       myText  += myCell + " + ";
     }
     localStorage.myText = myText;
    });
    $("#myBtnClear").click(function() {   //清空本地数据
     localStorage.clear();
     $("#myBtnRead").trigger("click");
  });})
 </script>
 <style type="text/css">
  td {background-color: lightblue; padding: 5px;}
  table {width: 400px;font-size: 14px; }
 </style></head>
<body><p><input type="button" value="从本地读取表格数据" id="myBtnRead"
style="width:131px">
   <input type="button" value="在本地保存表格数据" id="myBtnSave"
style="width:131px">
   <input type="button" value="清空本地数据" id="myBtnClear"
style="width:120px"></p><div id="div"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，localStorage 用于将页面上的指定数据保存到客户端本地的硬件设备中，即使浏览器被关闭该数据仍然存在，下次打开浏览器访问页面时仍然可以继续使用该数据；$('#myTable').prop("contenteditable", true)用于设置表格的每个单元格是可编辑的；localStorage.clear()可以清除此页面在本地硬盘中保存的所有内容。

此实例的源文件名是 myHtmlA075.html。

587 在本地保存文件时申请和查询磁盘配额

当在计算机中保存数据时，用户首先需要向计算机申请一定的磁盘配额(即当前应用可以使用的磁盘空间大小)，此实例主要使用 window.webkitStorageInfo.requestQuota()和 window.webkitStorageInfo.queryUsageAndQuota()方法实现在用户计算机中申请磁盘配额和查询磁盘配额。当在 Google Chrome 浏览器中显示该页面时，在"配额大小(bytes)："文本框中输入磁盘配额，以字节为单位，单击"申请磁盘配额"按钮，将弹出如图 587-1 所示的确认消息框，单击"允许"按钮，成功申请后将在弹出的消息框中显示当前应用可使用的磁盘配额，如图 587-2 所示；单击"查询磁盘配额"按钮，也将在弹出的消息框中显示当前应用可使用的磁盘配额。有关此实例的主要代码如下。

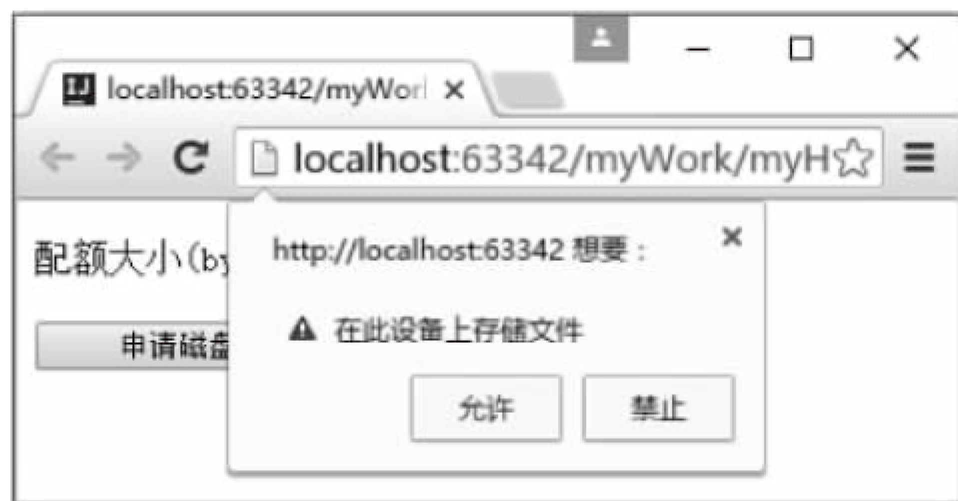


图 587-1

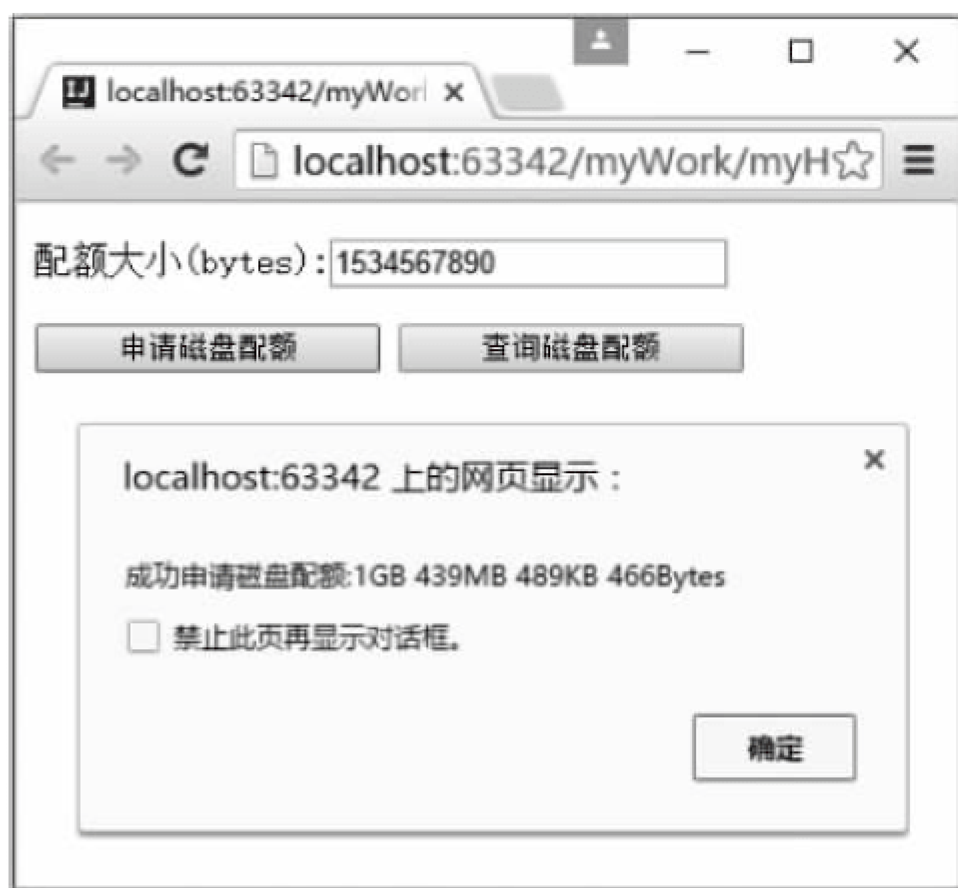


图 587-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function errorHandler(e) {                          //磁盘配额操作失败时执行的回调函数
   var msg='';
   switch(e.code) {
    case FileError.QUOTA_EXCEEDED_ERR:
     msg='文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
     break;
    case FileError.NOT_FOUND_ERR:
     msg='未找到文件或目录';
     break;
    case FileError.SECURITY_ERR:
     msg='操作不当引起安全性错误';
     break;
    case FileError.INVALID_MODIFICATION_ERR:
```

```
        msg = '对文件或目录所指定的修改处理不能被执行';
        break;
      case FileError.INVALID_STATE_ERR:
        msg = '指定的状态无效';
    }
    alert('当前操作引发错误:' + msg);
  }
  $(document).ready(function() {
   $("#myBtnGet").click(function() {       //申请磁盘配额
    var mySize = document.getElementById("mySize").value;
    window.webkitStorageInfo.requestQuota(PERSISTENT, mySize,
     function (grantedBytes) {             //申请磁盘配额成功时执行的回调函数
      var myInfo = "成功申请磁盘配额:"
      var strBytes, intBytes;
      if (grantedBytes >= 1024 * 1024 * 1024) {
        intBytes = Math.floor(grantedBytes/(1024 * 1024 * 1024));
        myInfo += intBytes + "GB ";
        grantedBytes = grantedBytes % (1024 * 1024 * 1024);
       }
       if (grantedBytes >= 1024 * 1024) {
        intBytes = Math.floor(grantedBytes/(1024 * 1024));
        myInfo += intBytes + "MB ";
        grantedBytes = grantedBytes % (1024 * 1024);
       }
       if (grantedBytes >= 1024) {
        intBytes = Math.floor(grantedBytes/1024);
        myInfo += intBytes + "KB ";
        grantedBytes = grantedBytes % 1024;
       }
       myInfo += grantedBytes + "Bytes";
       alert(myInfo); }, errorHandler);
    });
    $("#myBtnQuery").click(function() {  //查询磁盘配额
     window.webkitStorageInfo.queryUsageAndQuota(PERSISTENT,
      function (myUsage, myQuota) {        //查询磁盘配额信息成功时执行的回调函数
       var myInfo = "查询磁盘配额信息成功\n已用磁盘空间:"
       var strBytes, intBytes;
       if (myUsage >= 1024 * 1024 * 1024) {
        intBytes = Math.floor(myUsage/(1024 * 1024 * 1024));
        myInfo += intBytes + "GB ";
        myUsage = myUsage % (1024 * 1024 * 1024);
       }
       if (myUsage >= 1024 * 1024) {
        intBytes = Math.floor(myUsage/1024 * 1024);
        myInfo += intBytes + "MB ";
        myUsage = myUsage % 1024 * 1024;
       }
       if (myUsage >= 1024) {
        intBytes = Math.floor(myUsage/1024);
        myInfo += intBytes + "KB ";
        myUsage = myUsage % 1024;
       }
       myInfo += myUsage + "Bytes";
```

```
          myInfo += "\n 磁盘配额总空间: ";
          if (myQuota >= 1024 * 1024 * 1024) {
           intBytes = Math.floor(myQuota/(1024 * 1024 * 1024));
           myInfo += intBytes + "GB ";
           myQuota = myQuota % (1024 * 1024 * 1024);
         }
         if (myQuota >= 1024 * 1024) {
          intBytes = Math.floor(myQuota/(1024 * 1024));
          myInfo += intBytes + "MB ";
          myQuota = myQuota % (1024 * 1024);
         }
         if (myQuota >= 1024) {
          intBytes = Math.floor(myQuota/1024);
          myInfo += intBytes + "KB ";
          myQuota = myQuota % 1024;
         }
         myInfo += myQuota + "Bytes";
         alert(myInfo); }, errorHandler);
  });});
 </script></head>
<body><p><label>配额大小(bytes):<input id = "mySize" type = "text" maxlength = "20"
value = "1234567890"></label></p>
<p><input type = "button" value = "申请磁盘配额" id = "myBtnGet" style = "width:150px">
   <input type = "button" value = "查询磁盘配额" id = "myBtnQuery" style = "width:150px"> </p></body>
</html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，window. webkitStorageInfo. requestQuota()方法用于申请磁盘配额，该方法使用 4 个参数，第 1 个参数指定类型，可能值为 TEMPORARY 或 PERSISTENT，当值为 TEMPORARY 时表示为临时数据申请磁盘配额，当值为 PERSISTENT 时表示为永久数据申请磁盘配额。如果在用户计算机中保存临时数据，当用户计算机中其他磁盘空间不足时可能会删除此应用在磁盘配额中的数据。在磁盘配额中保存数据后，当浏览器被关闭或关闭计算机电源时这些数据不会丢失。第 2 个参数代表申请的磁盘空间的尺寸，单位为 byte。第 3 个参数值为一个函数，代表申请磁盘配额成功时执行的回调函数。第 4 个参数为一个 FileError 对象，存放申请磁盘配额失败时的各种错误信息。

window. webkitStorageInfo. queryUsageAndQuota()方法用于查询当前应用已经申请的磁盘配额，该方法也有 3 个参数，第 1 个参数指定类型，可能值为 TEMPORARY 或 PERSISTENT，当参数值为 TEMPORARY 时表示查询保存临时数据用的磁盘配额信息，当参数值为 PERSISTENT 时表示查询保存永久数据用的磁盘配额信息。第 2 个参数为一个函数，代表查询磁盘配额信息成功时执行的回调函数，在回调函数中可以使用两个参数，其中第 1 个参数值为磁盘配额中的已用磁盘空间尺寸，第 2 个参数值表示磁盘配额所指定的全部磁盘空间尺寸，单位为 byte。第 3 个参数为一个函数，代表查询磁盘配额信息失败时执行的回调函数，参数值为一个 FileError 对象，存放查询磁盘配额信息失败时的各种错误信息。

此实例的源文件名是 myHtmlA076. html。

588　在本地计算机中创建文件并读/写文件内容

此实例主要通过使用 getFile()方法执行文件操作，从而实现在本地计算机中创建文件并读/写文件内容。当在 Google Chrome 浏览器中显示该页面时，在“文件名称:”文本框中输入文件名称，例如

“newFile.txt”，在“文件大小(byte)：”文本框中输入文件尺寸，例如“1024”字节，单击“创建文件”按钮，如果此文件在本地计算机的沙箱中创建成功，则将在下面显示“写文件操作结束”，如图 588-1 所示。在“文件名称：”文本框中输入文件名称，例如“newFile.txt”，单击“读取文件”按钮，则将在弹出的消息框中显示 newFile.txt 文件在创建时预置的默认内容“这是在创建文件时写入的数据。”。如图 588-2 所示。在“文件名称：”文本框中输入文件名称，例如“newFile.txt”，在“文件内容：”文本框中输入内容，例如“这是正文”，单击“追加数据”按钮，则将在 newFile.txt 文件中插入新的内容，再次单击“读取文件”按钮，即可在弹出的消息框中看到追加的数据。即使关闭计算机，文件内容也保存在硬盘中。有关此实例的主要代码如下。

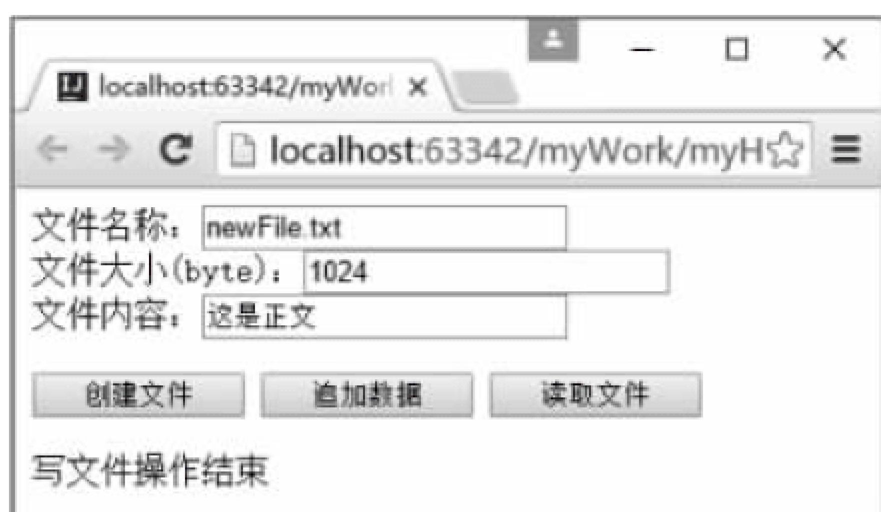


图　588-1



图　588-2

```
<!doctype html><html><head><meta charset=UTF-8>
<script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function createFile(){                //创建文件
   var mySize=document.getElementById("mySize").value;
   window.webkitRequestFileSystem(PERSISTENT,mySize,
    function(fs){                       //请求文件系统成功时所执行的回调函数
     var myName=document.getElementById("myName").value;
     fs.root.getFile(myName,{create: true},
      function(fileEntry) {
       fileEntry.createWriter(function(fileWriter) {
        fileWriter.onwriteend=function(e) {
         document.getElementById("myInfo").innerHTML='写文件操作结束';
        };
        fileWriter.onerror=function(e) {
         document.getElementById("myInfo").innerHTML='写文件操作失败: ';
        };
        var myData=new Blob(['这是在创建文件时写入的数据。']);
        fileWriter.write(myData); }, errorHandler);
       }, errorHandler);
      }, errorHandler );
    }
    function readFile(){                //读取文件
     var mySize=document.getElementById("mySize").value;
     window.webkitRequestFileSystem(PERSISTENT, mySize,
```

```
      function(fs){                       //请求文件系统成功时所执行的回调函数
       var myName = document.getElementById("myName").value;
       fs.root.getFile(myName, {create:false},
        function(fileEntry) {             //获取文件对象成功时所执行的回调函数
         fileEntry.file(
          function(file) {                //获取文件成功时所执行的回调函数
           var myReader = new FileReader();
           myReader.onloadend = function(e) {
           alert(this.result);
          };
         myReader.readAsText(file); },errorHandler);
       }, errorHandler);
     },errorHandler);
   }
   function addData(){                    //追加数据
    var mySize = document.getElementById("mySize").value;
    window.webkitRequestFileSystem(PERSISTENT,mySize,
     function(fs){                        //请求文件系统成功时所执行的回调函数
       var myName = document.getElementById("myName").value;
       fs.root.getFile(myName, {create:false},
        function(fileEntry) {
         fileEntry.createWriter(function(fileWriter) {
          fileWriter.onwriteend = function(e) {
        document.getElementById("myInfo").innerHTML = '向文件追加数据操作成功';
          };
          fileWriter.onerror = function(e) {
        document.getElementById("myInfo").innerHTML = '向文件追加数据操作失败：';
          };
         fileWriter.seek(fileWriter.length);
          var myData = new Blob([document.getElementById("myData").value]);
          fileWriter.write(myData);
         }, errorHandler); }, errorHandler);
       },errorHandler);
    }
    function errorHandler(e)
    {//请求文件系统失败时所执行的回调函数
      switch(e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
         msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
         break;
        case FileError.NOT_FOUND_ERR:
         msg = '未找到文件或目录';
         break;
        case FileError.SECURITY_ERR:
         msg = '操作不当引起安全性错误';
         break;
        case FileError.INVALID_MODIFICATION_ERR:
         msg = '对文件或目录所指定的操作不能被执行';
         break;
        case FileError.INVALID_STATE_ERR:
         msg = '指定的状态无效';
      };
      document.getElementById("myInfo").innerHTML = "当前操作引发错误：" + msg;
   }
 </script></head>
<body>文件名称：<input type = "text" id = "myName" value = "myFile.txt"><br/>
```

```
    文件大小(byte):<input type="text" id="mySize" value="1024"/><br/>
    文件内容:<input type="text" id="myData" value="这是正文"/><br/>
    <p><input type="button" value="创建文件" onclick="createFile()" style="width: 100px"><input
type="button" value="追加数据" onclick="addData()"style="width: 100px"><input type="button"
value="读取文件" onclick="readFile()"style="width: 100px"></p><output id="myInfo" ></output>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,getFile()方法用于访问文件,只有在访问成功后才能在其回调函数中通过 createWriter()方法创建文件和写入数据,通过 FileReader 读取文件内容。getFile()方法有 4 个参数,第 1 个参数是一个字符串值,代表需要创建或获取的文件名;第 2 个参数是一个自定义对象,在创建文件时必须将该对象的 create 值设置为 true,在读取文件时必须将该对象的 create 值设置为 false,在创建文件时如果文件已经存在,则覆盖该文件,如果该文件已经存在且被使用排他方式打开,则抛出错误;第 3 个参数是一个函数,表示读取文件或创建文件成功后执行的回调函数,在回调函数中可以使用一个参数,参数值是一个 FileEntry 对象,代表创建或读取成功的文件;第 4 个参数是一个函数,表示读取文件或创建文件失败时执行的回调函数,参数值是一个 FileError 对象,其中存放了读取文件或创建文件失败时的各种错误信息。

此实例的源文件名是 myHtmlA077.html。

589　将本地计算机中的多个文件复制到沙箱系统

此实例主要通过使用 FileWriter 对象的 write()方法实现将本地计算机的多个文件复制到沙箱系统中。当在 Google Chrome 浏览器中显示该页面时,单击“选择文件”按钮,然后在弹出的“打开”对话框中选择本地计算机的多个文件,如图 589-1 所示,单击“打开(O)”按钮返回,则选择的多个本地计算机文件将被复制到受浏览器保护的沙箱文件系统中,如图 589-2 所示;在“文件名称:”文本框中输入刚才复制的任意文件名称,例如“myHtmlA008.html”,单击“读取文件”按钮,则将在弹出的消息框中显示保存在沙箱中的 myHtmlA008.html 文件的内容。即使关闭计算机,复制的文件内容也保存在沙箱文件系统中。有关此实例的主要代码如下。

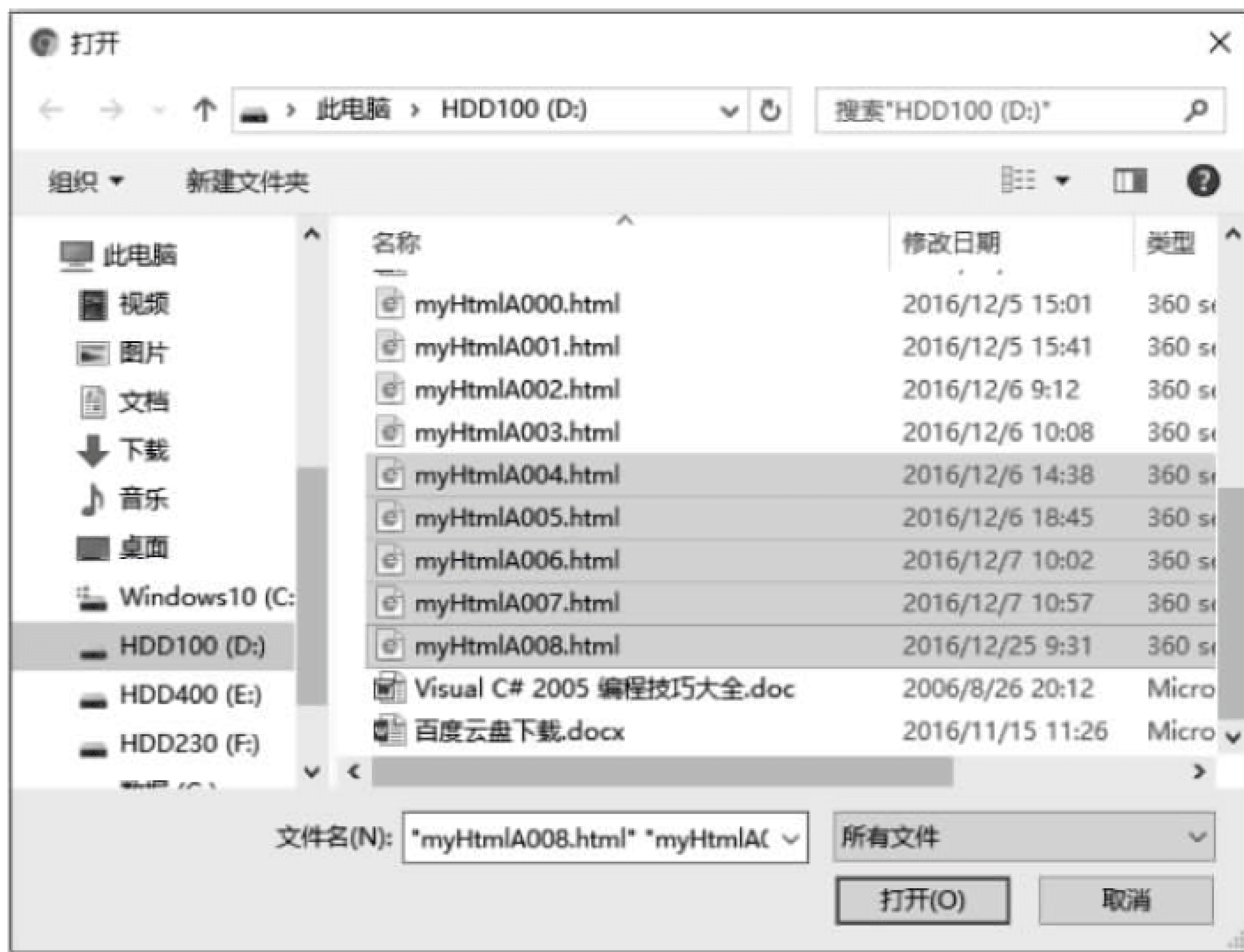


图　589-1

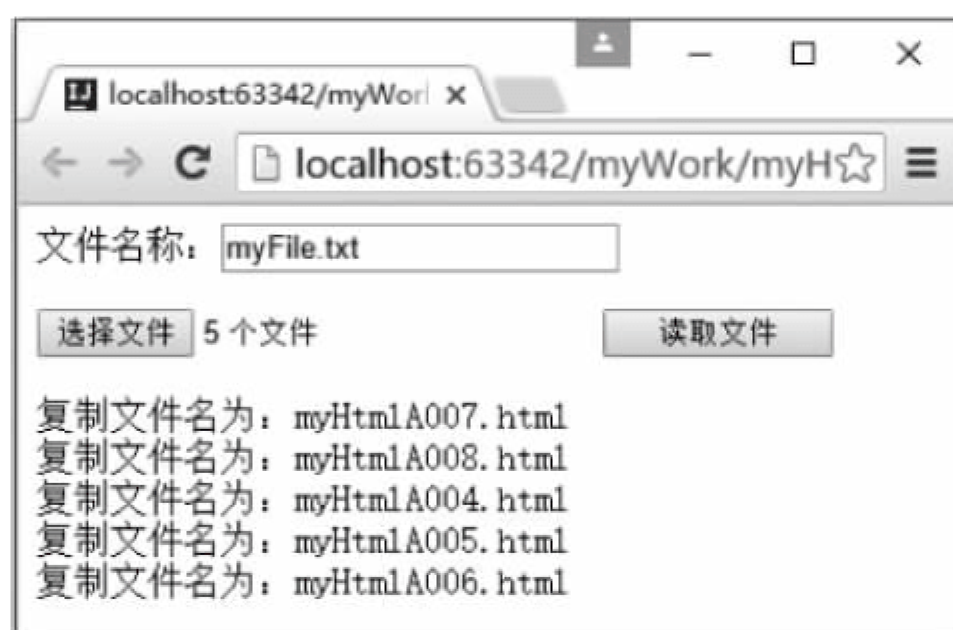


图　589-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function copyFile(){                  //复制文件
   var myFiles=document.getElementById("myFile").files;
   window.webkitRequestFileSystem(PERSISTENT,1024,
    function(fs){                       //请求文件系统成功时所执行的回调函数
     for(var i=0, myFile; myFile=myFiles[i]; ++i){
      (function(f) {
        fs.root.getFile(myFile.name, {create: true}, function(fileEntry) {
         fileEntry.createWriter(function(fileWriter) {
          fileWriter.onwriteend=function(e) {
           document.getElementById("myInfo").innerHTML+=
                                 '复制文件名为：'+f.name+'<br/>';
      };
      fileWriter.onerror=errorHandler
      fileWriter.write(f);
     }, errorHandler); }, errorHandler); })(myFile); }
   },errorHandler);
  }
  function readFile(){                  //读取文件
   window.webkitRequestFileSystem(PERSISTENT, 1024,
    function(fs){                       //请求文件系统成功时所执行的回调函数
     var myName=document.getElementById("myName").value;
     fs.root.getFile(myName, {create:false},
      function(fileEntry) {             //获取文件对象成功时所执行的回调函数
       fileEntry.file(
        function(file) {                //获取文件成功时所执行的回调函数
         var myReader=new FileReader();
         myReader.onloadend=function(e) {
           alert(this.result);
         };
         myReader.readAsText(file);
       },errorHandler);
     }, errorHandler);
    },errorHandler);
   }
   function errorHandler(e)
   {//请求文件系统失败时所执行的回调函数
    switch(e.code) {
```

```
        case FileError.QUOTA_EXCEEDED_ERR:
          msg='文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
          break;
        case FileError.NOT_FOUND_ERR:
          msg='未找到文件或目录';
          break;
        case FileError.SECURITY_ERR:
          msg='操作不当引起安全性错误';
          break;
        case FileError.INVALID_MODIFICATION_ERR:
          msg='对文件或目录所指定的操作不能被执行';
          break;
        case FileError.INVALID_STATE_ERR:
          msg='指定的状态无效';
      };
      document.getElementById("myInfo").innerHTML="当前操作引发错误:"+msg;
    }
  </script></head>
 <body>
  文件名称:<input type="text" id="myName" value="myFile.txt"><br/>
  <p><input type="file" id="myFile" onchange="copyFile()" multiple />
    <input type="button" value="读取文件" onclick="readFile()"
 style="width:100px"></p>
 <output id="myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,FileWriter 对象的 write()方法执行将复制的文件写入沙箱系统中的动作。在 FileSystem API 中,当需要将磁盘上的文件复制到受浏览器沙箱保护的文件系统中时,可以先使用 file 对象引用磁盘上的文件,然后将其写入文件系统中,例如 fileWriter.write(file)。FileWriter 对象的 write()方法只有一个参数,参数值是一个 Blob 对象,代表需要写入的二进制数据。在 HTML5 中,file 对象继承 Blob 对象,所以在 write()方法中可以使用 file 对象作为参数,表示使用某个文件中的原始数据进行写文件操作。

此实例的源文件名是 myHtmlA078.html。

590 删除受浏览器保护的沙箱系统中的指定文件

此实例主要通过使用 FileEntry 对象的 remove()方法实现删除本地计算机中受浏览器保护的沙箱文件系统中的指定文件。当在 Google Chrome 浏览器中显示该页面时,在"文件名称:"文本框中输入任意的文件名称,例如"myFile.txt",单击"创建文件"按钮,则将在沙箱系统中创建该文件;在"文件名称:"文本框中输入刚才创建的文件名称,例如"myFile.txt",单击"删除文件"按钮,则将从沙箱系统中删除该文件,如图 590-1 所示。如果在沙箱系统中不存在指定的文件,则单击"删除文件"按钮将报出错信息,如图 590-2 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function createFile(){                    //创建文件
   var mySize=document.getElementById("mySize").value;
   window.webkitRequestFileSystem(PERSISTENT,mySize,
    function(fs){                           //请求文件系统成功时所执行的回调函数
     var myName=document.getElementById("myName").value;
```

```
        fs.root.getFile(myName,{create: true},
         function(fileEntry) {
          fileEntry.createWriter(function(fileWriter) {
           fileWriter.onwriteend = function(e) {
            document.getElementById("myInfo").innerHTML = '写文件操作结束';
           };
           fileWriter.onerror = function(e) {
            document.getElementById("myInfo").innerHTML = '写文件操作失败: ';
           };
           var myData = new Blob(['这是在创建文件时写入的数据。']);
           fileWriter.write(myData);
          }, errorHandler);
         }, errorHandler);
       }, errorHandler );
     }
     function errorHandler(e)
     {//请求文件系统失败时所执行的回调函数
       switch(e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
         msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
         break;
        case FileError.NOT_FOUND_ERR:
         msg = '未找到文件或目录';
         break;
        case FileError.SECURITY_ERR:
         msg = '操作不当引起安全性错误';
         break;
        case FileError.INVALID_MODIFICATION_ERR:
         msg = '对文件或目录所指定的操作不能被执行';
         break;
        case FileError.INVALID_STATE_ERR:
         msg = '指定的状态无效';
       };
       document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
     }
     function deleteFile(){           //删除文件
      var mySize = document.getElementById("mySize").value;
      window.webkitRequestFileSystem(PERSISTENT,mySize,
       function(fs){                  //请求文件系统成功时所执行的回调函数
        var myName = document.getElementById("myName").value;
        fs.root.getFile(myName,{ create: false },
         function(fileEntry){         //获取文件成功时所执行的回调函数
          fileEntry.remove(           //删除文件成功时所执行的回调函数
         function() {
           document.getElementById("myInfo").innerHTML =
                                          fileEntry.name + '文件被成功删除';
         }, errorHandler); }, errorHandler);},errorHandler);
      }
    </script></head>
   <body>文件名称: <input type = "text" id = "myName" value = "myFile.txt"><br/>
     文件大小(byte): <input type = "text" id = "mySize" value = "1024"/><br/>
     <p><input type = "button" value = "创建文件" onclick = "createFile()"
   style = "width:150px">
       <input type = "button" value = "删除文件" onclick = "deleteFile()"
   style = "width:150px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，FileEntry 对象（代表一个文件）的 remove()方法使用两个参数，分别为删除文件成功时执行的回调函数和删除文件失败时执行的回调函数。即：

```
fileEntry.remove(
    function(){            //删除文件成功时执行的回调函数
    //代码略
    },errorHandler         //删除文件失败时执行的回调函数
);
```

此实例的源文件名是 myHtmlA079.html。

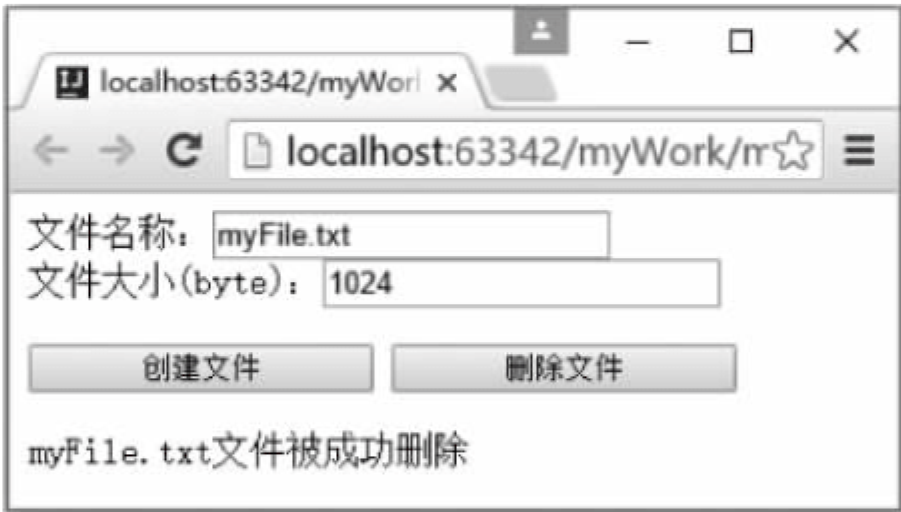


图　590-1

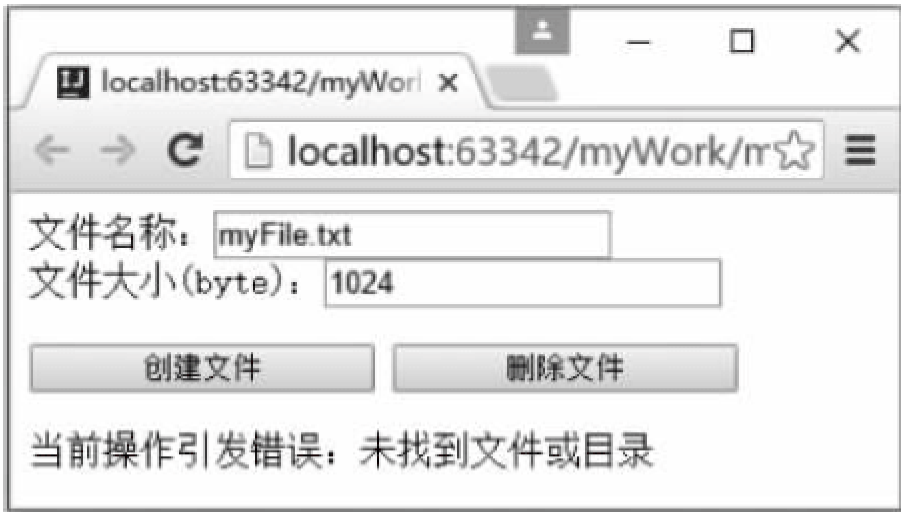


图　590-2

591　在本地沙箱文件系统中创建目录及其文件

此实例主要通过使用 getDirectory()方法实现在本地沙箱文件系统中创建目录并在该目录下创建文件及读取文件。当在 Google Chrome 浏览器中显示该页面时，在“目录名称：”文本框中输入目录名称，例如“myDirectory”，单击“创建目录”按钮，如果此目录在本地计算机的沙箱中创建成功，则将在下面显示目录完整路径及目录名等信息，如图 591-1 所示。在“文件名称：”文本框中输入文件名称，例如“newFile.txt”，在“文件大小(byte)：”文本框中输入文件尺寸，例如“1024”字节，单击“创建文件”按钮，如果此文件在此目录中创建成功，则将在下面显示“写文件操作结束”。在“文件名称：”文本框中输入文件名称，例如刚才创建的文件“newFile.txt”，单击“读取文件”按钮，则将在弹出的消息框中显示 newFile.txt 文件在创建时预置的默认内容“这是在子目录创建文件时写入的数据。”，如图 591-2 所示。即使关闭计算机，文件内容也保存在硬盘中。有关此实例的主要代码如下。

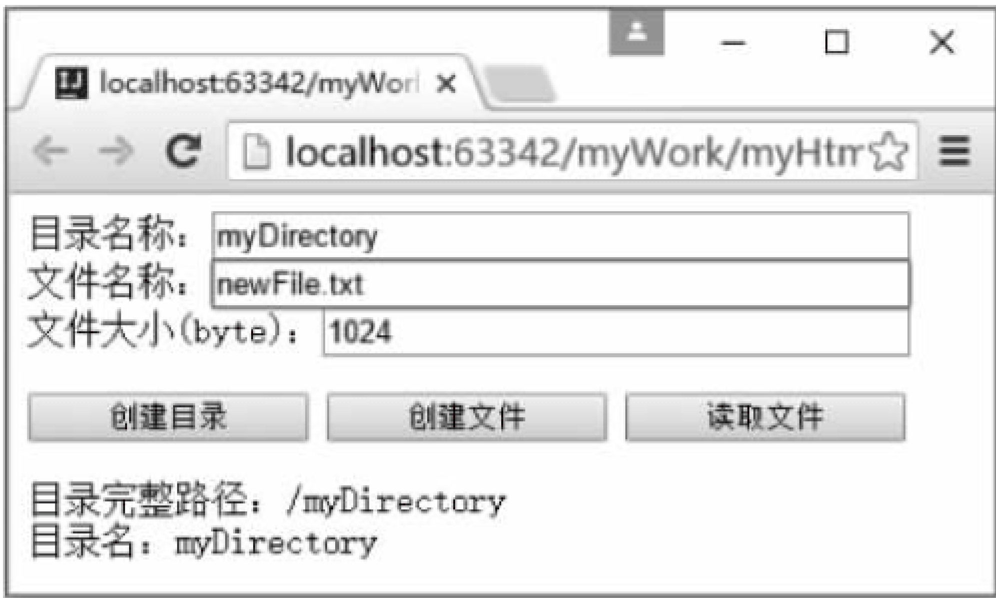


图　591-1

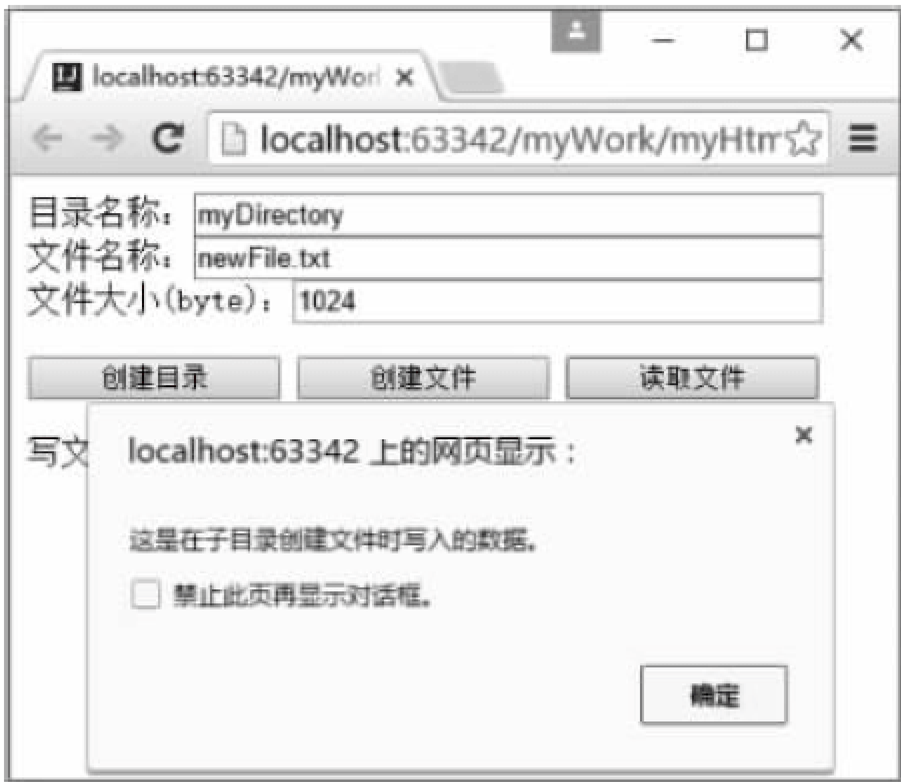


图　591-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function createDirectory(){                    //创建目录
   var mySize=document.getElementById("mySize").value;
   window.webkitRequestFileSystem(PERSISTENT,mySize,
    function(fs){                                //请求目录系统成功时所执行的回调函数
     var myDirectory=document.getElementById("myDirectory").value;
     fs.root.getDirectory(myDirectory,{ create: true },
      function(dirEntry){                        //创建目录成功时所执行的回调函数
       var myInfo="目录完整路径: "+dirEntry.fullPath+"<br>";
       myInfo+="目录名: "+dirEntry.name+"<br>";
       document.getElementById("myInfo").innerHTML=myInfo;
      }, errorHandler); }, errorHandler);
   }
   function createFile(){                        //创建文件
    var mySize=document.getElementById("mySize").value;
    var myDirectory=document.getElementById("myDirectory").value;
    window.webkitRequestFileSystem(PERSISTENT,mySize,
     function(fs){                               //请求文件系统成功时所执行的回调函数
      var myName=document.getElementById("myName").value;
      fs.root.getFile(myDirectory+"/"+myName,{create: true},
       function(fileEntry) {
        fileEntry.createWriter(function(fileWriter) {
         fileWriter.onwriteend=function(e) {
          document.getElementById("myInfo").innerHTML='写文件操作结束';
         };
         fileWriter.onerror=function(e) {
          document.getElementById("myInfo").innerHTML='写文件操作失败: ';
         };
         var myData=new Blob(['这是在子目录创建文件时写入的数据。']);
         fileWriter.write(myData);
       }, errorHandler); }, errorHandler); }, errorHandler );
    }
    function readFile(){                         //读取文件
     var mySize=document.getElementById("mySize").value;
     var myDirectory=document.getElementById("myDirectory").value;
     window.webkitRequestFileSystem(PERSISTENT, mySize,
      function(fs){                              //请求文件系统成功时所执行的回调函数
       var myName=document.getElementById("myName").value;
       fs.root.getFile(myDirectory+"/"+myName, {create:false},
         function(fileEntry) {                   //获取文件对象成功时所执行的回调函数
          fileEntry.file(
            function(file) {                     //获取文件成功时所执行的回调函数
              var myReader=new FileReader();
              myReader.onloadend=function(e) {
               alert(this.result);
              };
        myReader.readAsText(file); },errorHandler); }, errorHandler);
       },errorHandler);
    }
    function errorHandler(e)
    {//请求文件系统失败时所执行的回调函数
     switch(e.code) {
```

```
        case FileError.QUOTA_EXCEEDED_ERR:
         msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
         break;
        case FileError.NOT_FOUND_ERR:
         msg = '未找到文件或目录';
         break;
        case FileError.SECURITY_ERR:
         msg = '操作不当引起安全性错误';
         break;
        case FileError.INVALID_MODIFICATION_ERR:
         msg = '对文件或目录所指定的操作不能被执行';
         break;
        case FileError.INVALID_STATE_ERR:
         msg = '指定的状态无效';
       };
       document.getElementById("myInfo").innerHTML = "当前操作引发错误：" + msg;
    }
  </script></head>
 <body>目录名称：<input type = "text" id = "myDirectory" value = "myDirectory"
 style = "width:300px"><br/>
 文件名称：<input type = "text" id = "myName" value = "myFile.txt"
 style = "width:300px"><br/>
 文件大小(byte)：<input type = "text" id = "mySize" value = "1024"
 style = "width:252px"/><br/>
 <p><input type = "button" value = "创建目录" onclick = "createDirectory()"
  style = "width:122px">
        <input type = "button" value = "创建文件" onclick = "createFile()"
 style = "width:122px">
        <input type = "button" value = "读取文件" onclick = "readFile()"
 style = "width:122px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，getDirectory()方法用于创建子目录。在 FileSystem API 中，可以使用 DirectoryEntry 对象（代表文件系统中的一个目录）的 getDirectory()方法在一个目录中创建或获取子目录。getDirectory()方法有 4 个参数，第 1 个参数是一个字符串值，代表需要创建或获取的子目录名；第 2 个参数是一个自定义对象，当创建目录时必须将该对象的 create 属性值设定为 true，当获取目录时必须将该对象的 create 属性值设定为 false；第 3 个参数是一个函数，代表获取子目录或创建子目录成功时执行的回调函数，在回调函数中可以使用一个参数，参数值是一个 DirectoryEntry 对象，代表成功创建或获取的子目录；第 4 个参数值是一个函数，代表获取子目录或创建子目录失败时执行的回调函数，参数值是一个 FileError 对象，其中存放了获取子目录或创建子目录失败时的各种错误信息。

此实例的源文件名是 myHtmlA080.html。

592　在沙箱系统中使用递归函数创建多级目录

此实例主要使用递归函数调用 getDirectory()方法，从而实现在本地沙箱文件系统中创建多级目录。当在 Google Chrome 浏览器中显示该页面时，在“多级目录名称：”文本框中输入多级目录名称，例如“重庆市/渝北区/宝圣湖街道”，单击“创建多级目录”按钮，如果此多级目录在本地计算机的沙箱文件系统中创建成功，则将在下面显示目录完整路径及目录名等信息，如图 592-1 所示。在“文件名

称："文本框中输入文件名称，例如"myFile.txt"，在"文件大小(byte)："文本框中输入文件尺寸，例如"1024"字节，单击"创建文件"按钮，如果此文件在此多级目录的最底层目录中创建成功，则将在下面显示"写文件操作结束"。在"文件名称："文本框中输入文件名称，例如刚才创建的文件"myFile.txt"，单击"读取文件"按钮，则将在弹出的消息框中显示 myFile.txt 文件在创建时预置的默认内容"这是在子目录创建文件时写入的数据。"，即使关闭计算机，文件内容也保存在硬盘中。有关此实例的主要代码如下。



图　592-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function createDirectory(){                          //创建多级目录
   var myPath=document.getElementById("myDirectory").value;
    window.webkitRequestFileSystem(PERSISTENT, 1024,
     function(fs){                                     //请求文件系统成功时所执行的回调函数
      //使用递归函数创建每一级子目录
      createDir(fs.root, myPath.split('/'));
     },errorHandler);
   }
   function createDir(rootDirEntry, folders){   //创建多级目录时使用的递归函数
    //将"/foo/.//bar"之类的目录名中的'./'或'/'去除
    if (folders[0]=='.'|| folders[0]=='') {
       folders=folders.slice(1);
    }
    rootDirEntry.getDirectory(folders[0], {create: true},
     function(dirEntry) {                              //创建目录成功时所执行的回调函数
      if (folders.length) {
        var myInfo="目录完整路径："+dirEntry.fullPath+"<br>";
        myInfo+="目录名："+dirEntry.name+"目录已创建<br/>";
        document.getElementById("myInfo").innerHTML+=myInfo;
        //调用递归函数创建该目录下的子目录
        createDir(dirEntry, folders.slice(1));
      }
     }, errorHandler);
   }
   function createFile(){                              //(在最底层的目录中)创建文件
   }
   function readFile(){                                //(在最底层的目录中)读取文件
```

```
    }
    function errorHandler(e)
    {//请求文件系统失败时所执行的回调函数
     switch(e.code) {
        case FileError.QUOTA_EXCEEDED_ERR:
         msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
         break;
        case FileError.NOT_FOUND_ERR:
         msg = '未找到文件或目录';
         break;
        case FileError.SECURITY_ERR:
         msg = '操作不当引起安全性错误';
         break;
        case FileError.INVALID_MODIFICATION_ERR:
         msg = '对文件或目录所指定的操作不能被执行';
         break;
        case FileError.INVALID_STATE_ERR:
         msg = '指定的状态无效';
      };
      document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
    }
</script></head>
<body>多级目录名称: <input type = "text" id = "myDirectory" value = "重庆市/渝北区/宝圣湖街道" style =
"width:268px"><br/>
文件名称: <input type = "text" id = "myName" value = "myFile.txt"
style = "width:300px"><br/>
文件大小(byte): <input type = "text" id = "mySize" value = "1024"
style = "width:252px"/><br/>
   <p><input type = "button" value = "创建多级目录" onclick = "createDirectory()"
style = "width:155px">
      <input type = "button" value = "创建文件" onclick = "createFile()"
style = "width:105px">
     <input type = "button" value = "读取文件" onclick = "readFile()"
style = "width:105px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，自定义递归函数function createDir(rootDirEntry, folders)主要用于实现分级创建多级目录。因为在创建类似“重庆市/渝北区/宝圣湖街道”这种多级目录时，如果文件系统中不存在“渝北区”目录，则直接创建该目录下的“宝圣湖街道”目录，将会抛出错误，所以实例采用递归创建多级目录的方式来确保多级目录的层级关系，以避免发生错误。

此实例的源文件名是 myHtmlA081.html。

593 获取沙箱根目录下的子目录及其文件

此实例主要通过使用 DirectoryEntry 对象的 readEntries()方法实现获取沙箱根目录下的子目录及其文件。当在 Google Chrome 浏览器中显示该页面时，在“目录名称:”文本框中输入目录名称，例如“myDirectory”，单击“创建目录”按钮，如果此目录在本地计算机的沙箱中创建成功，则将在下面显示目录完整路径及目录名等信息。在“文件名称:”文本框中输入文件名称，例如“newFile.txt”，在“文件大小(byte):”文本框中输入文件尺寸，例如“1024”字节，单击“创建文件”按钮，如果文件在此目录

中创建成功，则将在下面显示“写文件操作结束”。单击“读取目录内容”按钮，则将在下面显示沙箱根目录下的所有文件和目录，包括刚才创建的目录和文件，如图 593-1 所示。即使关闭计算机，文件也保存在硬盘中。有关此实例的主要代码如下。

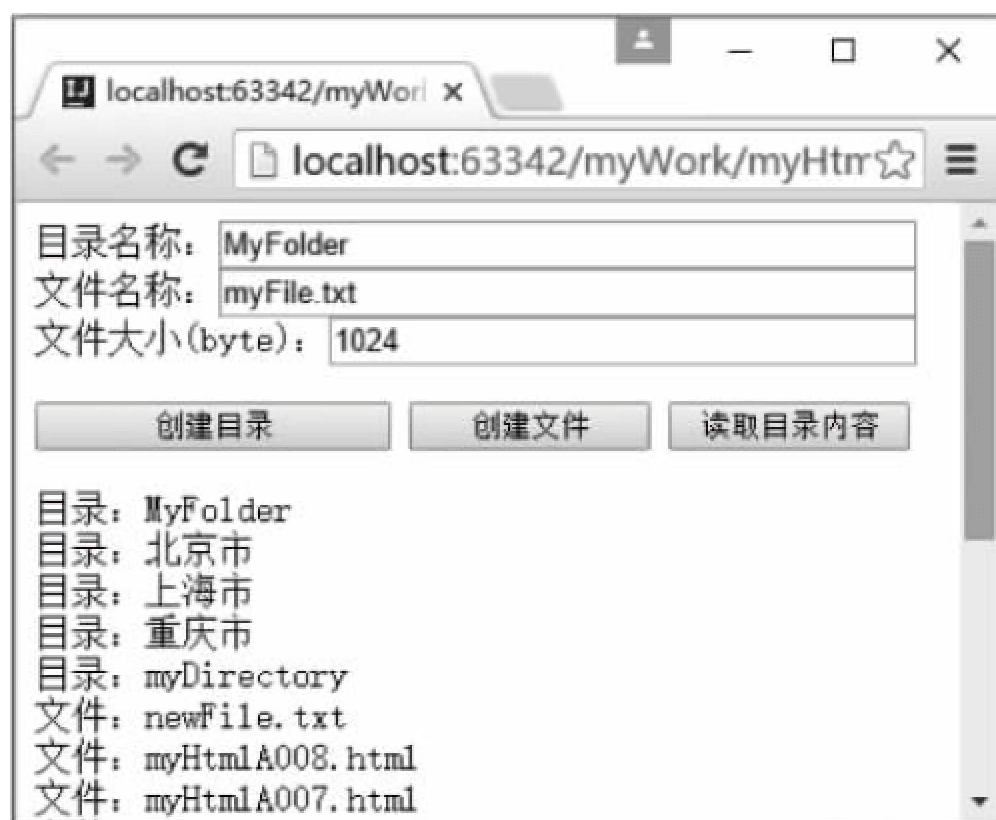


图 593-1

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function readDirectory(){                         //读取目录内容
   window.webkitRequestFileSystem(PERSISTENT,1024,
    function(fs){                                   //请求文件系统成功时所执行的回调函数
     var myReader=fs.root.createReader();
     var myEntries=[];
     //多次调用 readEntries()方法直到不再读出目录或文件
     var readEntries=function() {
      myReader.readEntries (
       function(results) {                          //读取目录成功时执行的回调函数
        if (!results.length) {
         listResults(myEntries.sort());
        }
        else {
         myEntries=myEntries.concat(toArray(results));
         readEntries();
       } },errorHandler);
       }; readEntries(); }, errorHandler);
    }
    function listResults(entries) {
     var type;
     entries.forEach(function(entry, i) {
      if(entry.isFile)
         type="文件: "+entry.name;
      else
         type="目录: "+entry.name;
      document.getElementById("myInfo").innerHTML+=type+"<br/>";
    }); }
    function toArray(list) {
     return Array.prototype.slice.call(list || [], 0);
    }
```

```
        function errorHandler(e)
        {//请求文件系统失败时所执行的回调函数
         switch(e.code) {
          case FileError.QUOTA_EXCEEDED_ERR:
           msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
           break;
          case FileError.NOT_FOUND_ERR:
           msg = '未找到文件或目录';
           break;
          case FileError.SECURITY_ERR:
           msg = '操作不当引起安全性错误';
           break;
          case FileError.INVALID_MODIFICATION_ERR:
           msg = '对文件或目录所指定的操作不能被执行';
           break;
          case FileError.INVALID_STATE_ERR:
           msg = '指定的状态无效';
         };
         document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
       }
       function createDirectory(){           //创建目录
       }
       function createFile(){                //创建文件
       }
    </script></head>
    <body>目录名称: <input type = "text" id = "myDirectory" value = "MyFolder"
    style = "width:300px"><br/>
    文件名称: <input type = "text" id = "myName" value = "myFile.txt"
    style = "width:300px"><br/>
    文件大小(byte): <input type = "text" id = "mySize" value = "1024"
    style = "width:252px"/><br/>
    <p><input type = "button" value = "创建目录" onclick = "createDirectory()"
    style = "width:155px">
            <input type = "button" value = "创建文件" onclick = "createFile()"
    style = "width:105px">
            <input type = "button" value = "读取目录内容" onclick = "readDirectory()"
    style = "width:105px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,DirectoryEntry 对象的 readEntries()方法用于读取目录的内容。在 FileSystem API 中读取目录的时候首先需要使用DirectoryEntry 对象的 createReader()方法创建DirectoryReader 对象,例如var myReader=fs.root.createReader()。在创建 DirectoryReader 对象之后就可以使用该对象的 readEntries()方法读取目录内容。readEntries()方法有两个参数,分别代表读取目录成功时执行的回调函数和读取目录失败时执行的回调函数。

```
myReader.readEntries(
function(results){//读取目录成功时执行的回调函数
//解析目录内容
},errorHandler//读取目录失败时执行的回调函数
);
```

此实例的源文件名是 myHtmlA082.html。

594 在沙箱文件系统中删除目录及其文件

此实例主要通过使用 DirectoryEntry 对象的 removeRecursively()方法实现在本地沙箱文件系统中删除目录及该目录下的所有子目录和文件。当在 Google Chrome 浏览器中显示该页面时，在“目录名称:”文本框中输入目录名称，例如“MyFolder”，单击“创建目录”按钮，如果此目录在本地计算机的沙箱中创建成功，则将在下面显示目录完整路径及目录名等信息。单击“读取目录内容”按钮，即可显示根目录下的所有目录和文件，包括刚才创建的目录“MyFolder”，如图 594-1 所示。在“目录名称:”文本框中输入目录名称，例如“MyFolder”，单击“删除目录”按钮，如果此目录在本地计算机的沙箱中被成功删除，单击“读取目录内容”按钮，即可显示根目录下的所有目录和文件，但不包括刚才删除的目录“MyFolder”，如图 594-2 所示。有关此实例的主要代码如下。

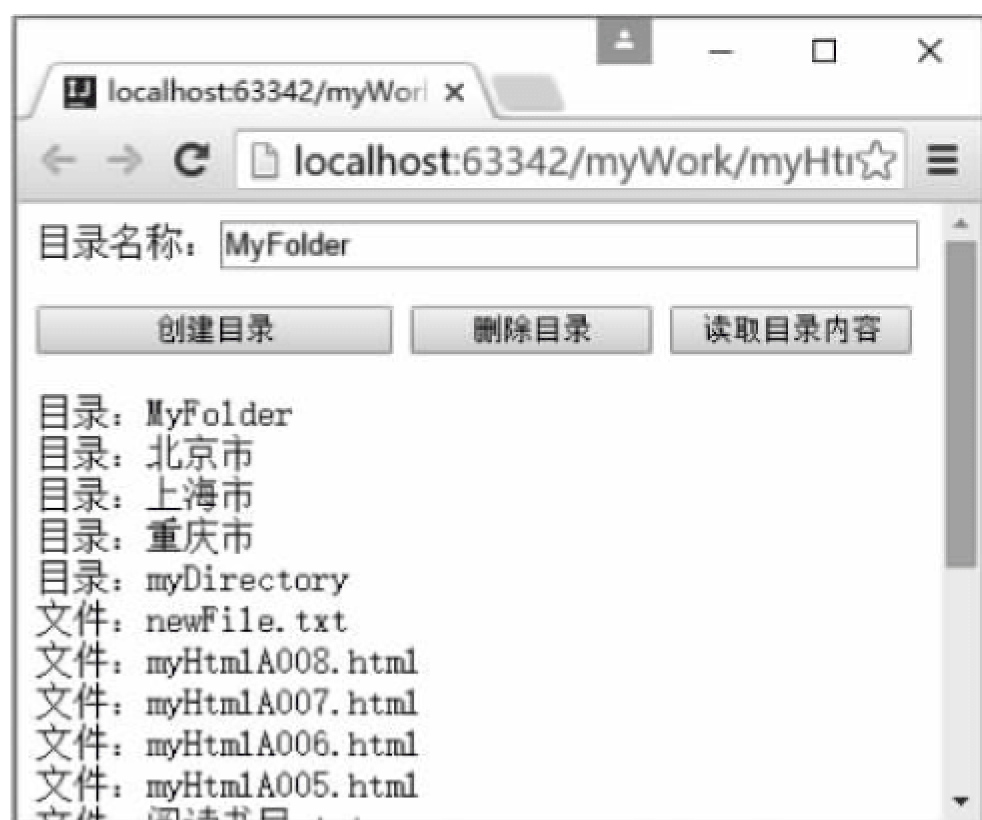


图 594-1



图 594-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  function deleteDirectory(){                    //删除目录
   window.webkitRequestFileSystem(PERSISTENT,1024,
    function(fs){                                //请求文件系统成功时所执行的回调函数
     var myDirectory=document.getElementById("myDirectory").value;
      fs.root.getDirectory(myDirectory, { create: false },
       function(dirEntry){                       //获取目录成功时所执行的回调函数
        dirEntry.removeRecursively(
         function() {                            //删除目录成功时所执行的回调函数
          document.getElementById("myInfo").innerHTML=
                                   dirEntry.name+'目录被删除';
         },
         //删除目录失败时所执行的回调函数
        errorHandler);
      },errorHandler); },errorHandler);
  }
  function errorHandler(e)
  {//请求文件系统失败时所执行的回调函数
   switch(e.code) {
    case FileError.QUOTA_EXCEEDED_ERR:
```

```
msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
          break;
         case FileError.NOT_FOUND_ERR:
          msg = '未找到文件或目录';
          break;
         case FileError.SECURITY_ERR:
          msg = '操作不当引起安全性错误';
          break;
         case FileError.INVALID_MODIFICATION_ERR:
          msg = '对文件或目录所指定的操作不能被执行';
          break;
         case FileError.INVALID_STATE_ERR:
          msg = '指定的状态无效';
        };
        document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
       }
       function createDirectory(){                    //创建目录
       }
     function readDirectory(){                        //读取目录内容
       }
     </script></head>
    <body>目录名称: <input type = "text" id = "myDirectory" value = "MyFolder"
    style = "width:300px"><br/>
    <p><input type = "button" value = "创建目录" onclick = "createDirectory()"
    style = "width:155px">
          <input type = "button" value = "删除目录" onclick = "deleteDirectory()"
    style = "width:105px">
          <input type = "button" value = "读取目录内容" onclick = "readDirectory()"
    style = "width:105px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，DirectoryEntry 对象的 removeRecursively()方法用于删除目录及其文件。removeRecursively()方法包括两个参数，分别表示删除目录成功时执行的回调函数和删除目录失败时执行的回调函数。

```
dirEntry.removeRecursively(
 function() {//删除目录成功时所执行的回调函数
  //显示删除结果
 },errorHandler//删除目录失败时所执行的回调函数
);
```

DirectoryEntry 对象的 remove()方法具有与 removeRecursively()方法类似的目录删除功能，但是 remove()方法只适用于删除空目录这种情形，否则会抛出错误。

此实例的源文件名是 myHtmlA083.html。

595 在沙箱系统中实现目录之间的文件复制

此实例主要通过使用 copyTo()方法实现在本地沙箱文件系统中将文件从一个目录复制到另一个目录。为了达到更加明晰的测试效果，首先在 Google Chrome 浏览器中打开源代码中的文件"myHtmlA080.html"，在"目录名称:"文本框中输入"北京市"，单击"创建目录"按钮，则将在本地计算机的沙箱中创建"北京市"目录，如图 595-1 所示。在"文件名称:"文本框中输入文件名称，例如

“newFile.txt”，在“文件大小(byte)：”文本框中输入文件尺寸，例如“1024”字节，单击“创建文件”按钮，如果此文件在“北京市”目录中创建成功，则将在下面显示“写文件操作结束”。在“目录名称：”文本框中输入“上海市”，单击“创建目录”按钮，则将在本地计算机的沙箱中创建“上海市”目录。然后在 Google Chrome 浏览器中打开实例程序源代码文件“myHtmlA084.html”，在“源目录名称：”文本框中输入“北京市”，在“目标目录名称：”文本框中输入“上海市”，在“文件名称：”文本框中输入“newFile.txt”，单击“复制文件”按钮，则“newFile.txt”文件将从“北京市”目录复制到“上海市”目录，如图 595-2 所示。最后在 Google Chrome 浏览器中打开源代码中的文件“myHtmlA080.html”，在“目录名称：”文本框中输入“上海市”，在“文件名称：”文本框中输入“newFile.txt”，单击“读取文件”按钮，则会显示刚才从“北京市”目录复制到“上海市”目录中的文件“newFile.txt”的内容，如图 595-3 所示。有关此实例的主要代码如下。

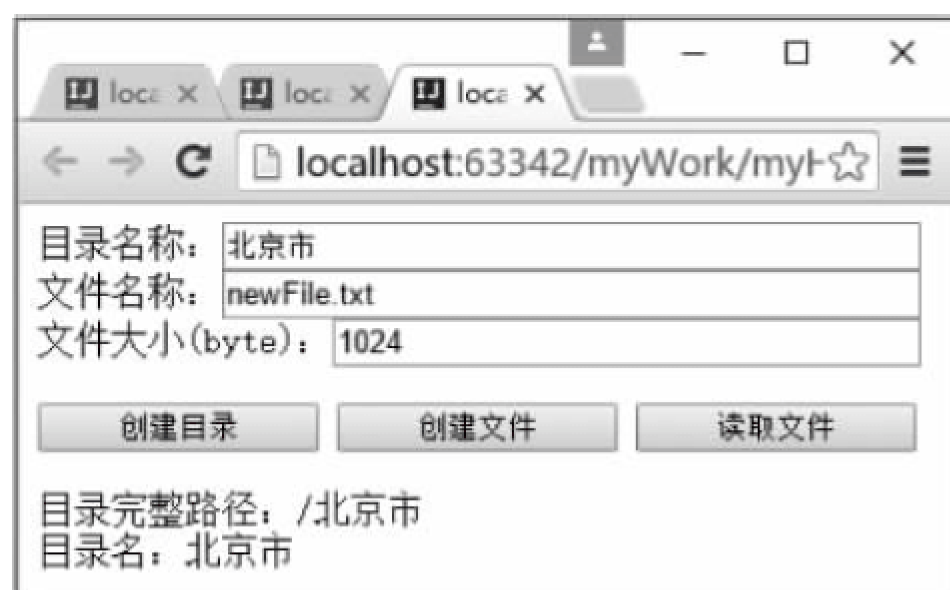


图 595-1

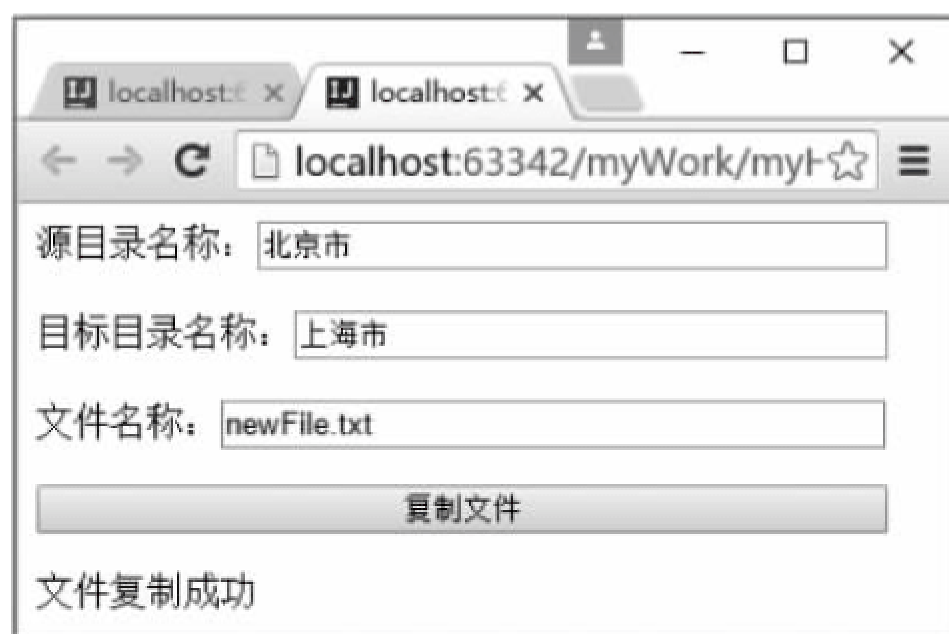


图 595-2



图 595-3

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  window.requestFileSystem =
             window.requestFileSystem || window.webkitRequestFileSystem;
  function copyFile(){                //复制文件
   var mySource = document.getElementById("mySource").value;
   var myTarget = document.getElementById("myTarget").value;
```

```
var myFile = document.getElementById("myFile").value;
window.requestFileSystem(window.PERSISTENT, 1024 * 1024, function(fs) {
  copy(fs.root, mySource + '/' + myFile, myTarget + '/');
}, errorHandler);
}
function copy(cwd, src, dest) {
 cwd.getFile(src, {create:false},
  function(fileEntry) {                    //获取被复制文件成功时执行的回调函数
   cwd.getDirectory(dest, {create:false},
    function(dirEntry) {                   //获取复制目标目录成功时执行的回调函数
     fileEntry.copyTo(dirEntry,fileEntry.name,
      //复制文件操作成功时执行的回调函数
      function() {
       document.getElementById("myInfo").innerHTML = '文件复制成功';
      },
      //复制文件操作失败时执行的回调函数
      errorHandler );},
      //获取复制目标目录失败时执行的回调函数
       errorHandler); }, errorHandler);
}
function errorHandler(e)
{//请求文件系统失败时所执行的回调函数
  switch(e.code) {
   case FileError.QUOTA_EXCEEDED_ERR:
    msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
    break;
   case FileError.NOT_FOUND_ERR:
    msg = '未找到文件或目录';
    break;
   case FileError.SECURITY_ERR:
    msg = '操作不当引起安全性错误';
    break;
   case FileError.INVALID_MODIFICATION_ERR:
    msg = '对文件或目录所指定的操作不能被执行';
    break;
   case FileError.INVALID_STATE_ERR:
    msg = '指定的状态无效';
  };
  document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
}
</script></head>
<body>源目录名称: <input type = "text" id = "mySource" style = "width:270px"><br/><br/>
目标目录名称: <input type = "text" id = "myTarget" style = "width:254px"><br/><br/>
文件名称: <input type = "text" id = "myFile" style = "width:285px"><br/>
<p><input type = "button" value = "复制文件" onclick = "copyFile()"
style = "width:370px"></p><output id = "myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,fileEntry对象(代表一个文件)的copyTo()方法用于将一个目录中的文件或子目录复制到另一个目录中。该方法使用4个参数,第1个参数是一个DirectoryEntry对象,用于指定将文件或目录复制到哪个目标目录中,DirectoryEntry对象代表该目标目录;第2个、第3个及第4个参数均为可选参数,第2个参数是一个字符串,用于指定复制后的文件名或目录名;第3个参数与第4个参数均为函数,代表复制成功后执行的回调函数及

复制失败后执行的回调函数。

此实例的源文件名是 myHtmlA084. html。

596 在沙箱文件系统中重命名指定的文件

此实例主要通过使用 moveTo()方法实现在本地沙箱文件系统中重命名指定的文件。为了达到更加明晰的测试效果，首先在 Google Chrome 浏览器中打开源代码中的文件“myHtmlA080. html”，在“目录名称：”文本框中输入“myDirectory”，单击“创建目录”按钮，则将在本地计算机的沙箱中创建“myDirectory”目录，如图 596-1 所示。在“文件名称：”文本框中输入文件名称，例如“OldFile. txt”，在“文件大小(byte)：”文本框中输入文件尺寸，例如“1024”字节，单击“创建文件”按钮，如果此文件在“myDirectory”目录中创建成功，则将在下面显示“写文件操作结束”。然后在 Google Chrome 浏览器中打开实例程序源代码文件“myHtmlA085. html”，在“目录名称：”文本框中输入“myDirectory”，在“原文件名称：”文本框中输入“OldFile. txt”，在“新文件名称：”文本框中输入“NewFile. txt”，单击“重命名文件”按钮，则“OldFile. txt”文件将被更名为“NewFile. txt”，如图 596-2 所示。最后在 Google Chrome 浏览器中打开源代码中的文件“myHtmlA080. html”，在“目录名称：”文本框中输入“myDirectory”，在“文件名称：”文本框中输入“NewFile. txt”，单击“读取文件”按钮，则会显示刚才更名之后的新文件“NewFile. txt”的内容，如图 596-3 所示。有关此实例的主要代码如下。

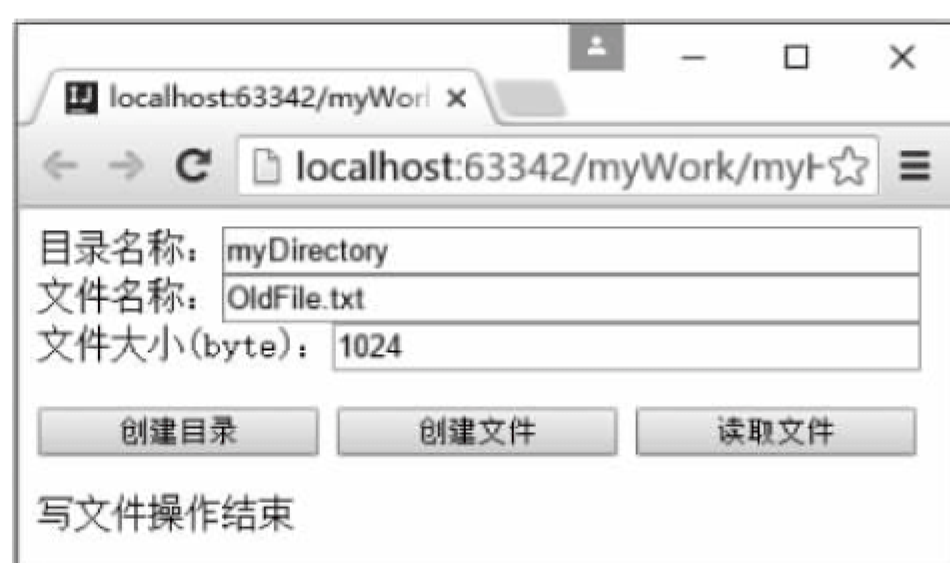


图 596-1

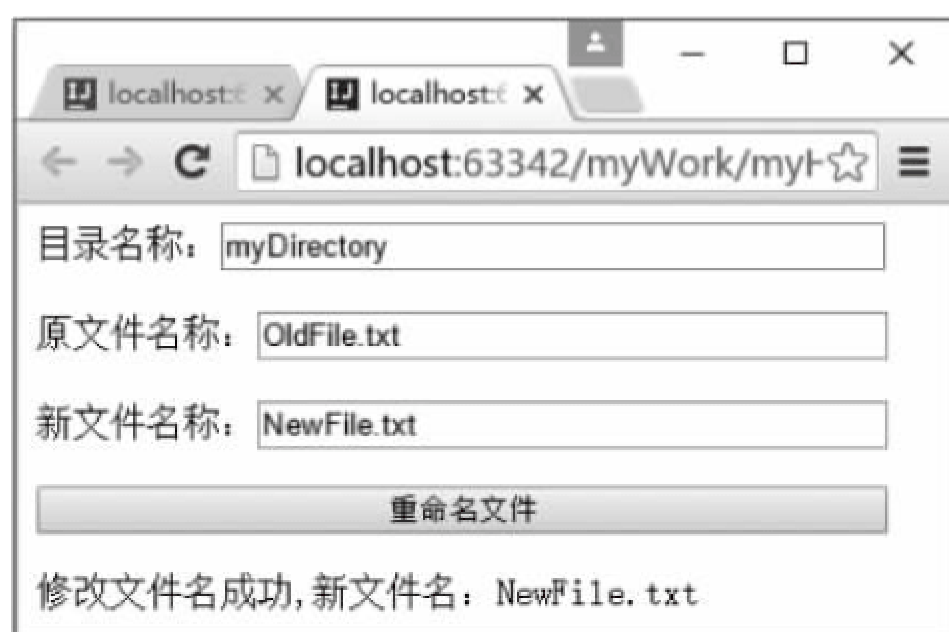


图 596-2

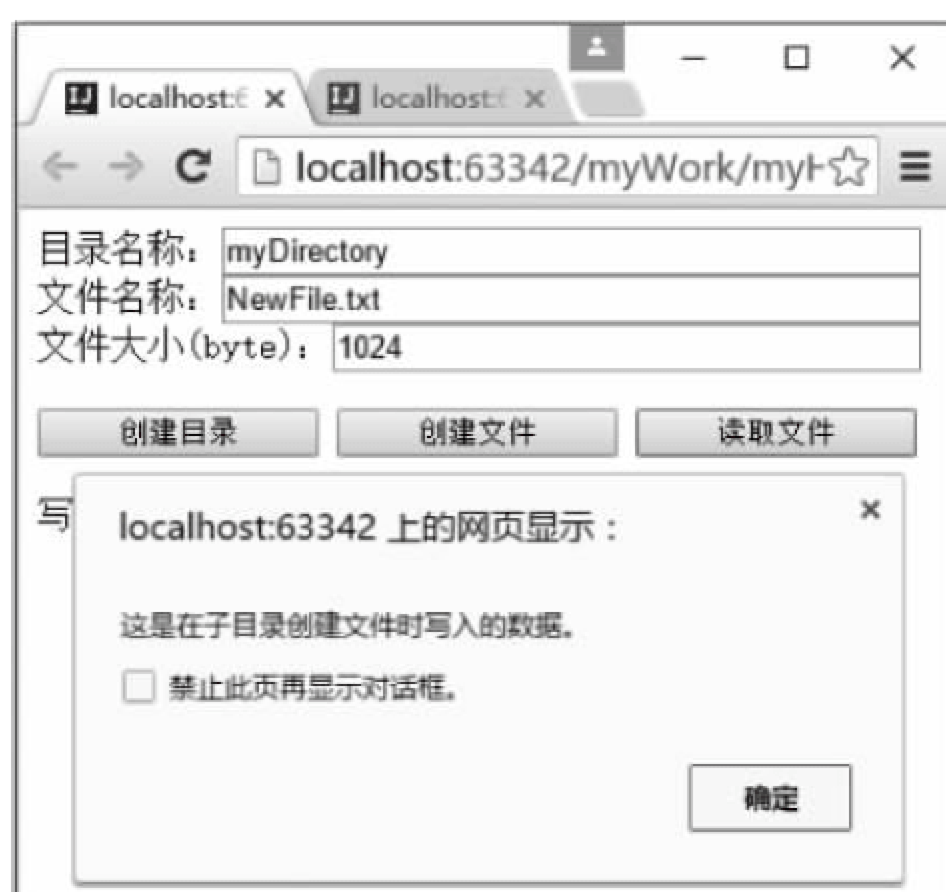


图 596-3

```
<!doctype html><html>
<head>
 <meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script>
 <script language="javascript">
  window.requestFileSystem =
          window.requestFileSystem || window.webkitRequestFileSystem;
  function renameFile(){                //重命名文件
   var myFolder = document.getElementById("myDirectory").value;
   var myOldFile = document.getElementById("mySource").value;
   var myNewFile = document.getElementById("myTarget").value;
   window.requestFileSystem(window.PERSISTENT, 1024 * 1024, function(fs) {
     rename(fs.root, myFolder + '/' + myOldFile,myNewFile,myFolder + '/');
   }, errorHandler);
  }
  function rename(cwd,oldFileName,newFileName,folder) {
   cwd.getFile(oldFileName, {create:false},
    function(fileEntry) {               //获取文件成功时执行的回调函数
     cwd.getDirectory(folder, {create:false},
      //获取文件目录成功时执行的回调函数
      function(folder) {
        fileEntry.moveTo(folder,newFileName,
         function() {                   //文件重命名操作成功时执行的回调函数
           document.getElementById("myInfo").innerHTML =
                        '修改文件名成功,新文件名: ' + newFileName;
          },errorHandler                //文件重命名操作失败时执行的回调函数
        );},errorHandler                //获取目录失败时执行的回调函数
           ); }, errorHandler           //获取文件失败时执行的回调函数
   );}
  function errorHandler(e)
  {//请求文件系统失败时所执行的回调函数
    switch(e.code) {
     case FileError.QUOTA_EXCEEDED_ERR:
      msg = '文件系统所使用的存储空间的尺寸超过磁盘限额控制中指定的空间尺寸';
      break;
     case FileError.NOT_FOUND_ERR:
      msg = '未找到文件或目录';
      break;
     case FileError.SECURITY_ERR:
      msg = '操作不当引起安全性错误';
      break;
     case FileError.INVALID_MODIFICATION_ERR:
      msg = '对文件或目录所指定的操作不能被执行';
      break;
     case FileError.INVALID_STATE_ERR:
      msg = '指定的状态无效';
   };
   document.getElementById("myInfo").innerHTML = "当前操作引发错误: " + msg;
  }
 </script>
</head>
<body>
目录名称: <input type="text" id="myDirectory" style="width:285px"><br/><br/>
```

```
原文件名称: <input type="text" id="mySource" style="width:270px"><br/><br/>
新文件名称: <input type="text" id="myTarget" style="width:270px"><br/>
<p><input type="button" value="重命名文件" onclick="renameFile()"
style="width:370px"></p><output id="myInfo" ></output></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，DirectoryEntry 对象(代表一个目录)的 moveTo()方法用于将一个目录中的文件或子目录复制到另一个目录中。但是由于可以在 moveTo()方法的第 2 个可选参数中重新指定移动后的文件名或目录名，所以在将移动源目录与目标目录指定为同一个目录并且指定一个新的文件名后即可实现重命名文件。moveTo()方法与 copyTo()方法所使用的参数及其说明完全相同，区别仅在于使用 copyTo()方法时将把指定文件或目录从复制源目录复制到目标目录中，复制后复制源目录中该文件或目录依然存在，但是在使用 moveTo()方法时将把指定文件或目录从移动源目录移动到目标目录中，移动后源目录中的该文件或目录被删除。

此实例的源文件名是 myHtmlA085.html。

第9部分 其他

597 使用@media查询创建响应式页面布局

此实例主要通过使用@media查询实现页面根据当前浏览器窗口的大小进行动态布局。当在Google Chrome浏览器中显示该页面时，如果浏览器窗口的最大宽度只有500px，则采用图597-1所示的布局，否则采用图597-2所示的布局。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  .header { padding:1.0121457489878542510121457489879%;
          background-color:#F1F1F1; border:1px solid #E9E9E9;}
  .menuitem { margin:4.310344827586206896551724137931%;margin-left:0; margin-top:0; padding:
4.310344827586206896551724137931%;border-bottom:1px solid #E9E9E9; cursor:pointer; }
  .main { padding:2.0661157024793388429752066115702%;}
  .right { padding:4.310344827586206896551724137931%;background-color:#CDF0F6;}
  h2{ border-bottom:1px solid black;}
  .footer {padding:1.0121457489878542510121457489879%; text-align:center; background-color:
#F1F1F1; border:1px solid #E9E9E9; font-size:0.625em;}
  .myContainer { width:100%; }
  .myWrapper { overflow:hidden; }
  .myBox { margin-bottom:2.0242914979757085020242914979757%;margin-right: 2.0242914979757085020
242914979757%; float:left;}
  .myHeader { width:100%; }
  .myMenu { width:23.481781376518218623481781376518%; }
  .myMain { width:48.987854251012145748987854251012%; }
  .myRight {width:23.481781376518218623481781376518%; margin-right:0;}
  .myFooter {width:100%; margin-bottom:0;}
  @media only screen and (max-width: 500px) {
   .myMenu { width:100%;}
   .menuitem { margin:1.0121457489878542510121457489879%;
             padding:1.0121457489878542510121457489879%; }
   .myMain { width:100%;}
   .main { padding:1.0121457489878542510121457489879%;}
   .myRight { width:100%; }
   .right { padding:1.0121457489878542510121457489879%;}
   .myBox { margin-right:0; float:left;} }
 </style></head>
<body><div class="myContainer">
```

```
<div class = "myWrapper">
  <div class = "myBox myHeader"><div class = "header"><h1>强东商城</h1></div></div>
  <div class = "myBox myMenu">
   <div class = "menuitem">商品介绍 </div><div class = "menuitem">规格参数</div>
   <div class = "menuitem">包装清单</div></div>
  <div class = "myBox myMain"><div class = "main"><h1>西莫寇纳格干红葡萄酒</h1><p>干红葡萄酒是指
葡萄酒酿造后,酿酒原料(葡萄汁)中的糖分完全转化成酒精,残糖量小于或等于 4.0g/L 的红葡萄酒。</p>
<img src = "img/B105.jpg"></div></div>
   <div class = "myBox myRight"><div class = "right"><h2>法国葡萄酒</h2><p>法国最古老的超一级酒庄
是吕萨吕斯酒堡。法国法律将法国葡萄酒分为 4 级:法定产区葡萄酒、优良地区餐酒、地区餐酒、日常餐酒。</p>
<h2>西班牙葡萄酒</h2><p>西班牙葡萄酒是被世界忽略的酒世界,更被很多中国人忽略。要知道,西班牙的
葡萄酒产量居世界第三位,而且有很多质量非常优秀的葡萄酒。</p></div></div>
   <div class = "myBox myFooter"><div class = "footer"><p>互联网出版许可证编号新出网证字 150 号  |
出版物经营许可证  |  网络文化经营许可证京网文 348 号  |  违法和不良信息举报电话: 4006561155 </p>
</div></div></div></div></body></html>
```



图 597-1



图 597-2

上面有底纹的代码是此实例的核心代码。在该部分代码中,@media only screen and (max-width: 500px)用于定义屏幕最大宽度为 500px 时的 CSS 样式。在 CSS3 中,使用@media 查询能够轻松实现针对不同的媒体类型定义不同的样式。特别是如果需要设置响应式的页面,@media 是非常有用的。在重置浏览器大小的过程中页面也会根据浏览器的宽度和高度重新渲染页面。@media 查询的语法格式如下。

```
@media  mediatype and|not|only (media feature) { CSS-Code;}
```

mediatype 媒体类型值的相关说明如表 597-1 所示。

表 597-1　mediatype 值的相关说明

值	说　明
all	用于所有设备
print	用于打印机和打印预览
screen	用于计算机屏幕、平板计算机、智能手机等
speech	应用于屏幕阅读器等发声设备

mediafeature 媒体功能值的相关说明如表 597-2 所示。

表 597-2　mediafeature 值的相关说明

值	说　明
aspect-ratio	定义输出设备中的页面可见区域宽度与高度的比率
color	定义输出设备每一组彩色原件的个数。如果不是彩色设备，则值等于 0
color-index	定义在输出设备的彩色查询表中的条目数。如果没有使用彩色查询表，则值等于 0
device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的比率
device-height	定义输出设备的屏幕可见高度
device-width	定义输出设备的屏幕可见宽度
grid	用来查询输出设备是否使用栅格或点阵
height	定义输出设备中的页面可见区域高度
max-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率
max-color	定义输出设备每一组彩色原件的最大个数
max-color-index	定义在输出设备的彩色查询表中的最大条目数
max-device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最大比率
max-device-height	定义输出设备的屏幕可见的最大高度
max-device-width	定义输出设备的屏幕最大可见宽度
max-height	定义输出设备中的页面最大可见区域高度
max-monochrome	定义在一个单色框架缓冲区中每像素包含的最大单色原件个数
max-resolution	定义设备的最大分辨率
max-width	定义输出设备中的页面最大可见区域宽度
min-aspect-ratio	定义输出设备中的页面可见区域宽度与高度的最小比率
min-color	定义输出设备每一组彩色原件的最小个数
min-color-index	定义在输出设备的彩色查询表中的最小条目数
min-device-aspect-ratio	定义输出设备的屏幕可见宽度与高度的最小比率
min-device-width	定义输出设备的屏幕最小可见宽度
min-device-height	定义输出设备的屏幕的最小可见高度
min-height	定义输出设备中的页面最小可见区域高度
min-monochrome	定义在一个单色框架缓冲区中每像素包含的最小单色原件个数
min-resolution	定义设备的最小分辨率
min-width	定义输出设备中的页面最小可见区域宽度
monochrome	定义在一个单色框架缓冲区中每像素包含的单色原件个数。如果不是单色设备，则值等于 0
orientation	定义输出设备中的页面可见区域高度是否大于或等于宽度
resolution	定义设备的分辨率。例如 96dpi、300dpi
scan	定义电视类设备的扫描工序
width	定义输出设备中的页面可见区域宽度

此实例的源文件名是 myHtmlB105.html。

598　使用@media 查询创建响应式表格布局

此实例主要通过使用@media 查询实现表格根据当前浏览器窗口的大小进行动态响应式布局。当在 Google Chrome 浏览器中显示该页面时，如果浏览器窗口的宽度超过 450px，则显示图 598-1 所示的表格；否则按照每列的数据以每行为组进行显示，如图 598-2 所示。有关此实例的主要代码如下。



图 598-1



图 598-2

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   td { background-color: lightseagreen; font-size: 13px; }
   th { background-color: seagreen; }
   table { border: 1px solid white;  width: 80%; margin: 0; padding: 0; border-collapse: collapse;
border-spacing: 0; margin: 0 auto;}
   table tr { border: 1px solid white; padding: 5px; }
   table th, table td { padding: 5px; text-align: center;
                        border: 1px solid white; }
   table th { font-size: 14px; letter-spacing: 1px; }
   @media screen and (max-width: 450px) {
     table { border: 0; }
     table thead { display: none; }
     table tr { margin-bottom: 10px; display: block;
                border-bottom: 0px solid #DDD; }
     table td { display: block; text-align: right; font-size: 13px;
                border-bottom: 0px dotted #CCC; }
     table td:last-child { border-bottom: 0; }
     table td:before { content: attr(myHead); float: left; font-weight: bold; } }
 </style></head>
<body><table><thead>
  <tr><th>北京市</th><th>上海市</th><th>天津市</th><th>重庆市</th></tr></thead>
  <tbody>
    <tr><td myHead="北京市">北京市公安局</td>
        <td myHead="上海市">上海市审计局</td>
        <td myHead="天津市">天津市人民检察院</td>
        <td myHead="重庆市">重庆市环保局</td></tr>
    <tr><td myHead="北京市">北京市人力社保局</td>
        <td myHead="上海市">上海市国土房管局</td>
        <td myHead="天津市">天津市监察局</td>
        <td myHead="重庆市">重庆市三峡水库管理局</td></tr>
```

```
    <tr><td myHead="北京市">北京市经济信息委员会</td>
        <td myHead="上海市">上海市地税局</td>
        <td myHead="天津市">天津市工商局</td>
    <td myHead="重庆市">重庆市规划局</td></tr></tbody></table></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，@media screen and (max-width: 450px){}表示在屏幕宽度大于450px时使用大括号内设置的表格样式。在这里screen表示媒体类型是显示屏幕，and被称为关键字，与其相似的还有not、only等，(max-width: 450px)表示媒体条件是屏幕宽度大于450px，当然还有(min-width: 450px)等其他表现形式。

此实例的源文件名是myHtmlB215.html。

599　使用png图像重置默认的鼠标指针形状

此实例主要通过使用png图像设置cursor属性实现重置默认的鼠标箭头指针形状。当在Google Chrome浏览器中显示该页面时，如果鼠标指针在图像上任意滑动，则默认的鼠标箭头指针将被左上角的电话所代替，即电话跟随鼠标移动，如图599-1所示。有关此实例的主要代码如下。



图　599-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  img { border-radius: 10px;}
  /*当鼠标指针悬浮在图像上时显示此定制光标*/
  img:hover { cursor: url('img/B066A.png'), default; }
 </style></head>
<body><img src="img/B152A.jpg"></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，img:hover { cursor: url('img/B066A.png'), default; }表示鼠标指针在图像上悬浮时使用B066A.png图像代替默认的鼠标箭头指针。在CSS中，cursor属性用于设置或检索在对象上移动的鼠标指针采用何种系统预定义的光标形状，该属性的语法格式如下。

```
cursor: [<url> [<x> <y>]?,] * [ auto | default | none | context-menu | help | pointer | progress | wait
| cell | crosshair | text | vertical-text | alias | copy | move | no-drop | not-allowed | e-resize | n-
resize | ne-resize | nw-resize | s-resize | se-resize | sw-resize | w-resize | ew-resize | ns-
resize | nesw-resize | nwse-resize | col-resize | row-resize | all-scroll]
```

上面的每个属性值代表具体的预置鼠标指针形状，例如 cursor：help 显示带问号的鼠标指针。当然也可以像实例这样使用图像来代替鼠标指针，需要注意的是，不同的浏览器支持的图像格式不同。

此实例的源文件名是 myHtmlB216.html。

600 在全屏显示模式下锁定和解锁鼠标指针

此实例主要通过使用 requestPointerLock()方法实现在全屏显示模式时锁定鼠标指针和解锁鼠标指针。当在 Google Chrome 浏览器中显示该页面时，单击“锁定鼠标指针”按钮，则浏览器自动给出提示信息“此网页现处于全屏模式并意图隐藏鼠标指针。”，如图 600-1 所示；单击“允许”按钮，则鼠标指针从屏幕上消失，并显示提示信息“此网页现处于全屏模式并已隐藏鼠标指针。按 Esc 键退出。”，如图 600-2 所示，此时在屏幕上看不见鼠标指针，如果按下 Esc 键，则退出全屏模式，解锁鼠标指针，鼠标指针恢复正常状态。此技术主要适用于网页游戏场景。有关此实例的主要代码如下。

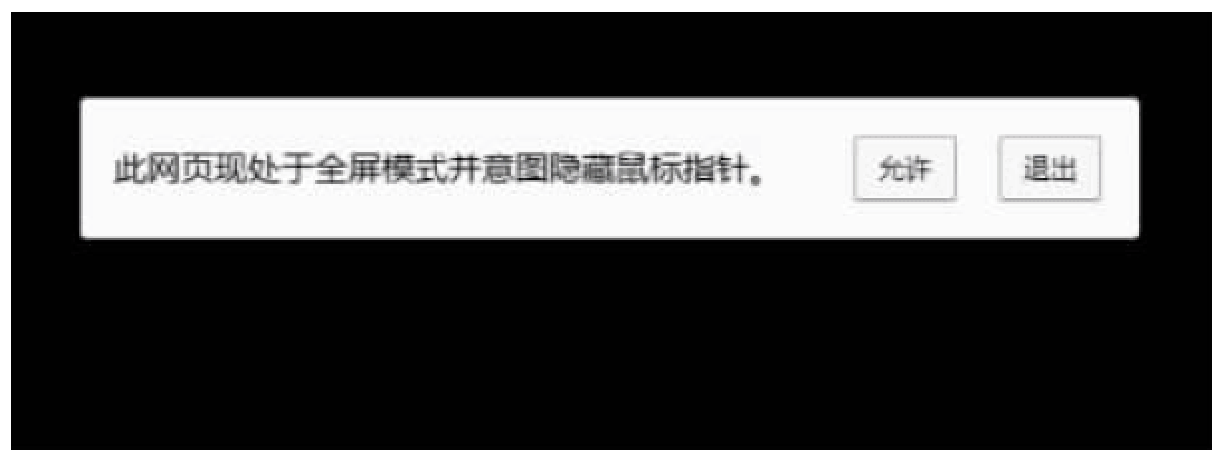


图 600-1

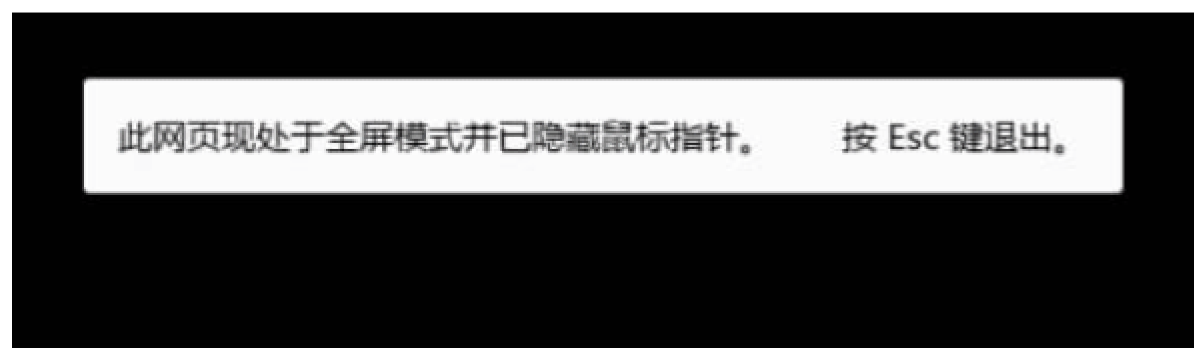


图 600-2

```
<!doctype html><html><head><meta charset=UTF-8>
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  var myDiv;
  function lockPointer() {
   myDiv=document.getElementById("myDiv");
   //将 div 块设为全屏状态
   //到目前为止在针对某个元素锁定鼠标指针之前首先要将该元素设为全屏状态
   myDiv.requestFullscreen=myDiv.requestFullscreen ||
      myDiv.mozRequestFullscreen ||myDiv.mozRequestFullScreen ||
      myDiv.webkitRequestFullscreen;
   myDiv.requestFullscreen();
  }
```

```
document.addEventListener("mousemove", function(e) {
  var movementX = e.movementX || e.mozMovementX || e.webkitMovementX || 0;
  var movementY = e.movementY || e.mozMovementY || e.webkitMovementY || 0;
  console.log("鼠标指针位置: movementX = " + movementX + ",movementY = " + movementY);
}, false);
document.addEventListener('fullscreenchange', fullscreenChange, false);
document.addEventListener('mozfullscreenchange', fullscreenChange, false);
document.addEventListener('webkitfullscreenchange', fullscreenChange, false);
function fullscreenChange() {
  if (document.webkitFullscreenElement === myDiv ||
      document.mozFullscreenElement === myDiv ||
      document.mozFullScreenElement === myDiv) {
      //元素已处于全屏状态,锁定鼠标指针
   myDiv.requestPointerLock = myDiv.requestPointerLock ||
    myDiv.mozRequestPointerLock || myDiv.webkitRequestPointerLock;
   myDiv.requestPointerLock();
} }
</script></head>
<body><button onclick = "lockPointer();" style = "width:350px">锁定鼠标指针</button>
<div id = "myDiv"></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,元素(例如 div 元素)的 requestPointerLock()方法用于锁定鼠标指针。当鼠标指针的锁定状态发生改变时,例如调用 requestPointerLock()方法成功锁定鼠标指针,或调用 exitPointerLock()方法成功取消鼠标指针锁定,或用户在全屏状态下按下 Esc 键,此时将触发 document 对象的 pointerlockchange 事件,该事件不携带任何数据。当调用 requestPointerLock()方法锁定鼠标指针操作失败时,或调用 exitPointerLock()方法取消鼠标指针锁定操作失败时,将触发 document 对象的 pointerlockerror 事件,该事件不携带任何数据。此外,不管鼠标指针是否处于锁定状态,都可以使用 movementX 属性与 movementY 属性来获取鼠标指针的位置信息,即使在鼠标指针没有被锁定的情况下。当鼠标指针没有被锁定时,可以被移动到浏览器窗口之外后再进入浏览器窗口,这时 movementX 属性值与 movementY 属性值都将被初始化为 0。

此实例的源文件名是 myHtmlA105.html。

601 全屏显示页面或使页面退出全屏模式

此实例主要通过使用 RequestFullScreen()方法和 CancelFullScreen()方法实现全屏显示页面或使页面退出全屏显示模式。当在 Google Chrome 浏览器中显示该页面时,单击"全屏显示网页"按钮,如图 601-1 所示,则页面立即切换到全屏显示状态;在全屏显示状态的页面中单击"退出全屏显示"按钮,如图 601-2 所示,则页面立即退出全屏显示模式。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset = UTF - 8>
  <style>
   html { background: url(img/a050A.jpg) no - repeat;
          min - height: 100 % ; background - size: 100 %  100 % ; }
</style></head>
<body><input type = "button" id = "myBtn" value = "全屏显示网页"
      onclick = "toggleFullScreen();" style = "width:250px;">
<p style = "width:100 % ;color:yellow;" id = "myInfo">当前状态: 网页是非全屏显示的</p>
```

```
<script language = "javascript">
    var myDocument = document.documentElement;
    var myInfo = document.getElementById("myInfo");
    var myBtn = document.getElementById("myBtn");
    document.addEventListener("webkitfullscreenchange", function () {
     myInfo.innerHTML = "当前状态: " + (document.webkitIsFullScreen) ? "网页是全屏显示的" : "网页是非
全屏显示的";
    myBtn.value = (document.webkitIsFullScreen)?"退出全屏显示":"全屏显示网页";}, false);
    function toggleFullScreen() {
        if (myBtn.value == "全屏显示网页") {
            if (myDocument.webkitRequestFullScreen) {
                myDocument.webkitRequestFullScreen();
        } } else {   if (document.webkitCancelFullScreen) {
                document.webkitCancelFullScreen(); } } }
</script></body></html>
```



图 601-1



图 601-2

上面有底纹的代码是此实例的核心代码。在该部分代码中，DOM 对象的根结点对象(document.documentElement)或某个元素的 RequestFullScreen(谷歌的 Chrome 浏览器是 webkitRequestFullScreen)属性值用于判断浏览器是否支持 FullScreen API，如果支持，则可以直接调用该元素的 RequestFullScreen()方法(谷歌的 Chrome 浏览器是 webkitRequestFullScreen()方法)实现将页面或元素设置为全屏显示状态。当然，也可以通过 DOM 对象的根结点对象(document.documentElement)或某个元素的 CancelFullScreen(谷歌的 Chrome 浏览器是 webkitCancelFullScreen)属性值或 exitFullScreen 属性值判断浏览器是否支持 FullScreen API，如果支持，则可以直接调用该元素的 CancelFullScreen()方法(谷歌的 Chrome 浏览器是 webkitCancelFullScreen()方法)实现将页面或元素设置为非全屏显示状态。

在 FullScreen API 中，可以通过监听 DOM 对象(document.documentElement)或某个元素的 fullscreenchange 事件(当页面或元素从非全屏状态进入全屏显示状态或者从全屏显示状态退出时触发，在代码中需要根据不同的浏览器添加前缀，谷歌的 Chrome 浏览器是 webkitfullscreenchange)并指定事件处理函数来指定当页面或元素变为全屏显示状态或者非全屏显示状态时所需要执行的代码。在事件处理函数中，可以通过 DOM 对象的根结点对象(document.documentElement)或某个元素的 FullScreen 属性值(在代码中需要根据不同的浏览器添加前缀，谷歌的 Chrome 浏览器是 webkitIsFullScreen，火狐浏览器是 mozFullScreen)来判断页面或元素是否处于全屏显示状态；如果该属性值为 true，则为全屏显示状态，否则为非全屏显示状态。

此实例的源文件名是 myHtmlA104.html。

602　创建自定义右键菜单代替默认右键菜单

此实例主要为 document 的 contextmenu 事件重新添加自定义方法，从而实现使用自定义右键菜单代替默认的右键菜单。当在 Google Chrome 浏览器中显示该页面时，使用鼠标右键单击空白页面将弹出自定义右键菜单，如图 602-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script type="text/javascript">
  document.oncontextmenu=function(e){          //使用鼠标右键单击页面显示自定义右键菜单
   if (window.event) e=window.event;
     var myMenu=document.getElementById("myMenu");
     myMenu.style.visibility="visible";        //显示自定义右键菜单
     myMenu.style.left=e.clientX+5+"px";       //设置自定义右键菜单的位置
     myMenu.style.top=e.clientY+5+"px";
     return false; }
  document.onclick=function() {                //使用鼠标单击页面的任意位置隐藏自定义右键菜单
   var myMenu=document.getElementById("myMenu");
   myMenu.style.visibility="hidden"; }
 </script>
 <style type="text/css">
  img { width: 400px; height: 250px; border-radius: 10px; margin: 2px;}
   * { margin: 0; padding: 0;}
   #myMenu { width: 130px; visibility: hidden; border-bottom-width:0; position: absolute;background:
lightgrey; padding-left: 5px; padding-right: 5px; box-shadow: 3px 5px 8px #888888; border-radius:
5px; }
   #myMenu li { list-style: none; text-indent: 10px; }
  .myNormal { display: block; height: 30px; line-height: 30px; border-bottom: 1px solid black; text-
decoration: none;  color: #666; font: 12px/30px tahoma; }
  .myLast { display: block; height: 30px; line-height: 30px; text-decoration: none;
color: black; font: 12px/30px tahoma;}
   #myMenu li a:hover { background: white; color: black;}
 </style></head>
<body><div id="myMenu">
 <ul><li><a href="#" class="myNormal">以滤色模式显示图像</a></li>
     <li><a href="#" class="myNormal">以增亮模式显示图像</a></li>
     <li><a href="#" class="myNormal">以差值模式显示图像</a></li>
     <li><a href="#" class="myNormal">以叠加模式显示图像</a></li>
     <li><a href="#" class="myNormal">以排除模式显示图像</a></li>
     <li><a href="#" class="myNormal">以色相模式显示图像</a></li>
     <li><a href="#" class="myLast">以混合模式显示图像</a></li></ul></div>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，document.oncontextmenu 表示文档 contextmenu 事件对应的方法，contextmenu 事件在用户右击鼠标时触发并打开元素的上下文菜单。如果没有对该方法进行自定义，则文档对象将执行默认的操作。如图 602-2 所示的右键菜单即是空白页面的默认右键菜单。

此实例的源文件名是 myHtmlA115.html。



图 602-1

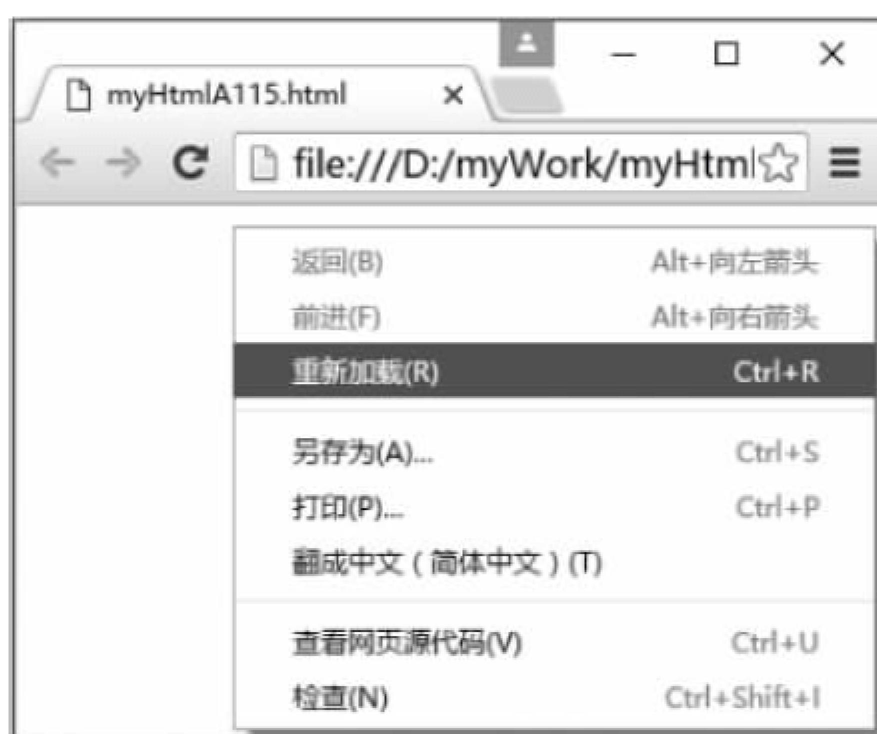


图 602-2

603 通过上下文菜单返回值禁止使用右键菜单

此实例主要设置页面 body 的 oncontextmenu()方法的返回值为 false，从而实现禁止在用户使用鼠标右键单击页面上的元素时弹出右键菜单。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标右键单击图像，则将弹出一个消息框，提示“此页面禁止使用右键菜单！”，如图 603-1 所示。有关此实例的主要代码如下。

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  img{ width:400px; height:250px; border-radius: 10px; }
 </style></head>
<body oncontextmenu="alert('此页面禁止使用右键菜单!'); return false;">
<p><img src="img/B315.jpg"></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<body oncontextmenu="alert('此页面禁止使用右键菜单!'); return false;">表示禁止在用户使用鼠标右键单击页面上的元素时弹出右键菜单，如果将 oncontextmenu()方法的返回值设置为 true，例如<body oncontextmenu=" return true;">，则当使用鼠标右键单击图像时将正常弹出右键菜单，如图 603-2 所示。

此实例的源文件名是 myHtmlB315.html。



图 603-1



图 603-2

604 使用字符编码解决中文字符的对齐问题

此实例主要通过使用字符编码   和   解决中文字符的对齐问题。当在 Google Chrome 浏览器中显示该页面时，"姓名："和"手机号："虽然与"电子邮箱："不等长，但是由于使用了字符编码   和   代替空格，因而实现了对齐，如图 604-1 所示。有关此实例的主要代码如下。

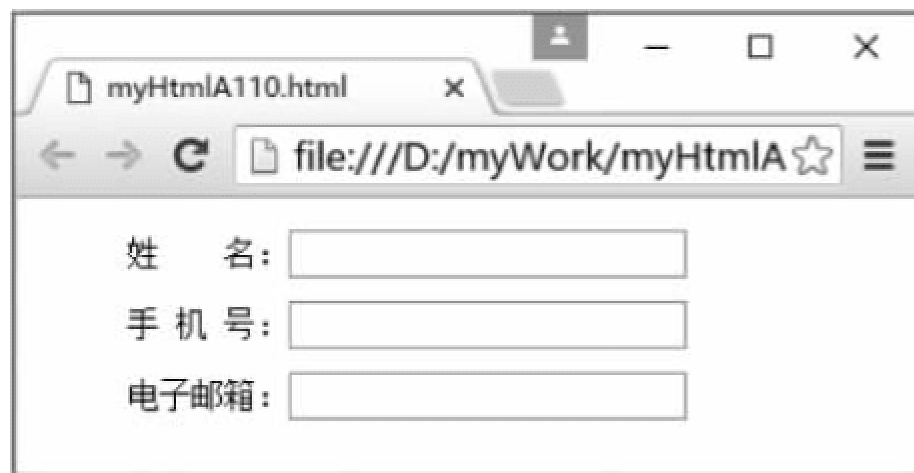


图 604-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <style>
  ul {list - style - type: none; font - size: 14px;}
  .li {margin - top: 10px; _font - family: '宋体'; }
 </style></head>
<body><ul><li class = "li">姓   名：<input type = "text"/></li>
 <li class = "li">手  机  号：<input type = "text"/></li>
 <li class = "li">电子邮箱：<input type = "text"/></li></ul></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，字符编码   和   代表两个空格符。实际上，所有的中文字符及特殊字符都有编码，都可以采用编码的方式来解决。例如，"<li class="li"> 电子邮箱：<input type="text"/>
"与"<li class="li">电子邮箱：<input type="text"/>
"的效果完全相同，即"电子邮箱"是中文字符"电子邮箱"的编码。

此实例的源文件名是 myHtmlA110.html。

605 实现十六进制编码和中文字符的相互转换

此实例主要通过使用 JavaScript 的 replace()方法实现十六进制编码和中文字符的相互转换。当在 Google Chrome 浏览器中显示该页面时，在文本框中输入"中国"，单击"将文本框内容转换成UNICODE"，如图 605-1 所示，则将显示"中国"对应的十六进制编码"中国"，如图 605-2 所示。反之，如果在文本框中输入"中国"，单击"将文本框内容还原为中文字符"按钮，如图 605-2 所示，则将显示"中国"编码对应的汉字"中国"，如图 605-1 所示。这主要用于在国际化的网页中采用十六进制编码解决汉字的乱码问题。有关此实例的主要代码如下。



图 605-1



图 605-2

```
<!doctype html><html><head><meta charset="UTF-8"></head>
<body><div align=center><p style=" width: 420px">在下列文本框中输入中文文字,单击"将文本框内容转换成 UNICODE"按钮,即将其转化为 UNICODE 编码。再单击"将文本框内容还原为中文字符"按钮,即将其还原为简体中文。</p></div>
<p align=center><textarea cols=55 rows=5 id=code>这是测试文字</textarea></p>
<p align=center>
 <input type=button onclick=encode(code,this) value="将文本框内容转换成 UNICODE" style="width:410px;">
 <script language="javascript">
  var mode="zhuan";
  function encode(obj, btn) {
   if (mode=="zhuan") {
    obj.value=obj.value.replace(/[^\u0000-\u00FF]/g, function ($0) {
      return escape($0).replace(/(%u)(\w{4})/gi, "&#x$2;")
    });
    btn.value="将文本框内容还原为中文字符";
      mode="huan";
    } else {
      obj.value=unescape(obj.value.replace(/&#x/g, '%u').replace(/;/g, ''));
      btn.value="将文本框内容转换成 UNICODE";
      mode="zhuan";
  } }
 </script></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中,replace()方法用于在字符串中用一些字符替换另一些字符,或替换一个与正则表达式匹配的子串,该方法的语法格式如下。

```
stringObject.replace(regexp/substr,replacement)
```

其中,参数 regexp/substr 规定子字符串或要替换的模式的 RegExp 对象,如果该值是一个字符串,则将它作为要检索的直接文本模式,而不是首先被转换为 RegExp 对象;参数 replacement 是一个字符串值,规定了替换文本或生成替换文本的函数。

replace()方法将在字符串中查找与 regexp 相匹配的子字符串,然后用 replacement 来替换这些子字符串。如果 regexp 具有全局标志 g,那么 replace()方法将替换所有匹配的子字符串,否则它只替换第一个匹配子字符串。replacement 可以是字符串,也可以是函数。如果它是字符串,那么每个匹配都将由字符串替换。

“中国”的 Unicode 字符编码是 U+4E2D 和 U+56FD,十六进制(&#x)表示的 4E2D 和 56FD 就是十进制(&#)的 20013 和 22269,所以 中国和 中国这两种写

法都会在页面显示时转换为“中国”二字，例如“< li class＝"li">中国< input type＝"text"/>”在网页上显示的就是中国和文本框。

此实例的源文件名是 myHtmlA111. html。

606　使用 meta 元信息实现页面的定时跳转与刷新

此实例主要通过设置 meta 元素的元信息 refresh 实现页面的定时跳转与刷新。当在 Google Chrome 浏览器中显示该页面时将提示“有事快干：5 秒后将跳转到京东首页！”，如图 606-1 所示，在经过 5 秒后页面自动跳转到京东首页。有关此实例的主要代码如下。



图　606-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <meta http - equiv = "refresh" content = "5; url = 'https://www.jd.com'">
</head>
<body><p><strong>有事快干: 5 秒后将跳转到京东首页!</strong></p></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，< meta http-equiv＝"refresh" content＝"5; url＝'https://www.jd.com'">表示页面在 5 秒后自动跳转到“https://www.jd.com”。如果需要使页面 5 秒自动刷新一次，则应该为< meta http-equiv＝"refresh" content＝"5; ">；如果需要使页面在 5 秒后自动跳转到首页，则应该为< meta http-equiv＝"refresh" content＝"5; url＝'/'">。在 HTML5 中，上述功能并不是 meta 元素的主要功能，< meta >元素的主要功能是提供有关页面的元信息（meta-information），例如针对搜索引擎和更新频度的描述和关键字。< meta >元素位于文档的头部，不包含任何内容。< meta >标签的属性定义了与文档相关联的名称/值对，这些名称/值对的作用如表 606-1 所示。

表 606-1　< meta >标签的属性定义的名称/值对

属　　性	值	说　　明
charset	character encoding	定义文档的字符编码
content	some_text	定义与 http-equiv 或 name 属性相关的元信息
http-equiv	content-type	把 content 属性关联到 HTTP 头部
	expires	
	refresh	
	set-cookie	
name	author	把 content 属性关联到一个名称
	description	
	keywords	
	generator	
	revised	
	others	

例如，使用< meta >元素定义针对搜索引擎的关键字。

```
<meta name = "keywords" content = "HTML, CSS, XML, XHTML, JavaScript" />,使用<meta>元素定义对页面的描述<meta name = "description" content = "免费的 Web 技术教程。" />
```

此实例的源文件名是 myHtmlA109. html。

607 实现单击网页的任意位置立即触发广告

此实例主要通过创建透明层实现单击网页的任意位置即可立即触发广告。当在 Google Chrome 浏览器中显示该页面时，如果试图首先在文本框中输入内容，则会触发广告跳转到百度首页；实际上，在首次显示该页面时单击页面的任何位置都将跳转到百度首页，如图 607-1 所示。有关此实例的主要代码如下。



图 607-1

```
<!doctype html><html><head><meta charset = "UTF - 8">
 <script src = "js/jquery - 3.1.1.min.js"></script>
 <style type = "text/css">
  /* 广告图片 */
  img{ position: relative; top:50 %; left:20 % }
  /* 透明层 */
  .layer { text - align: center; position: absolute;  top: 0; left: 0; margin: 0;
z - index: 1000; width: 100 %; height: 100 %;background - color: transparent; cursor:help;}
 </style></head>
<body><div class = "test"> 特别声明：本站需要广告支持维护，每次都会弹出广告。单击弹出广告，若再次回到本页，即可正常操作。<br/><br/>
    请输入搜索内容：<input type = "text"/> <button type = "submit">搜索网盘</button>
</div>
<div class = "layer" onclick = "window.open('http://www.baidu.com'); $(this).remove();"><img src =
"img/B067A.png"/></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，z-index：1000 表示透明层位于页面的最顶层。background-color：transparent 表示该层是透明的，因此用户感觉不到它的存在，用户看到的是下面底层的内容，由于透明层的阻隔，可见的底层内容是不可操作的。onclick＝"window.open('http://www.baidu.com')；$(this).remove()；"表示在单击此透明层之后跳转到百度首页，然后移除透明层，因此用户在单击透明层的广告后跳转到百度首页，然后返回此页面即是移除透明层之后的页面，故可以在文本框中输入内容。

此实例的源文件名是 myHtmlB199. html。

608 通过使用滤镜特效实现灰色主题的网页

此实例主要设置-webkit-filter 属性为 grayscale(100%)，即大家通常所说的灰色滤镜，从而实现以灰色主题显示网页。当在 Google Chrome 浏览器中显示该页面时，无论网易首页显示的元素和颜色多么复杂，在灰色滤镜的作用下一切都将变为灰色，如图 608-1 所示。有关此实例的主要代码如下。



图 608-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  html{ -webkit-filter:grayscale(100%);}
 </style></head>
<body><iframe src="http://www.163.com" width="450" height="300"></iframe>
</body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，html{ -webkit-filter: grayscale(100%);}表示对 html 元素对应的类中的所有内容使用灰色滤镜，在此实例中，iframe{-webkit-filter:grayscale(100%);}与此具有相同的效果。在 CSS3 中，filter 属性可以在元素呈现之前为元素的渲染提供一些效果，例如模糊、颜色转移之类，filter 属性常用于调整图像、背景、边框的渲染。filter 属性的语法格式如下。

```
-webkit-filter:none|blur(px)|brightness()|contrast()|grayscale()|hue-rotate(deg)|invert()|opacity()
|saturate()|sepia()|drop-shadow(radius)|url()
```

其中，各属性值的说明如下。

(1) none：无 SVG 滤镜效果，此为默认值。

(2) blur(<number>px)：设置对象的模糊效果。

(3) brightness(<percentage>)：设置对象的亮度。除了百分比外，也可以用非负数表达。

(4) contrast(<percentage>)：设置对象的对比度。除了百分比外，也可以用 0～1 的数字表达。

(5) grayscale(<percentage>)：设置对象的灰度。除了百分比外，也可以用 0～1 的数字表达。

(6) hue-rotate(<number>deg)：设置对象的色相旋转。用 0～360 数字表达。

(7) invert(<percentage>)：设置对象的反色。除了百分比外，也可以用 0～1 的数字表达。

(8) opacity(<percentage>)：设置对象的透明度。除了百分比外，也可以用 0～1 的数字表达。

（9）saturate(< percentage >)：设置对象的饱和度。除了百分比外，也可以用非负数表达。

（10）sepia(< percentage >)：设置对象的褐色程度。除了百分比外，也可以用 0～1 的数字表达。

（11）drop-shadow(< length > < length > radius < color >)：设置对象的阴影。第 1 个长度是向右偏移距离；第 2 个长度是向下偏移距离，皆可为负值，皆为必传参数；第 3 个参数是阴影圆角，可选；第 4 个参数是阴影颜色，可选。

（12）url(path. svg＃a)：设置对象滤镜效果。

通过 SVG 可实现以上所有效果。SVG 可写在页面里，也可在外部引用，可增加锚点。

此实例的源文件名是 myHtmlB198. html。

609　使用 MutationObserver 监视 DOM 变化

此实例主要通过使用 MutationObserver(突变监视器)监视 DOM 变化。当在 Google Chrome 浏览器中显示该页面时，单击“插入一幅图像来测试 DOM 变化”按钮，则 MutationObserver(突变监视器)将立即在上面显示监测的 DOM 变化信息，如图 609-1 所示。有关此实例的主要代码如下。

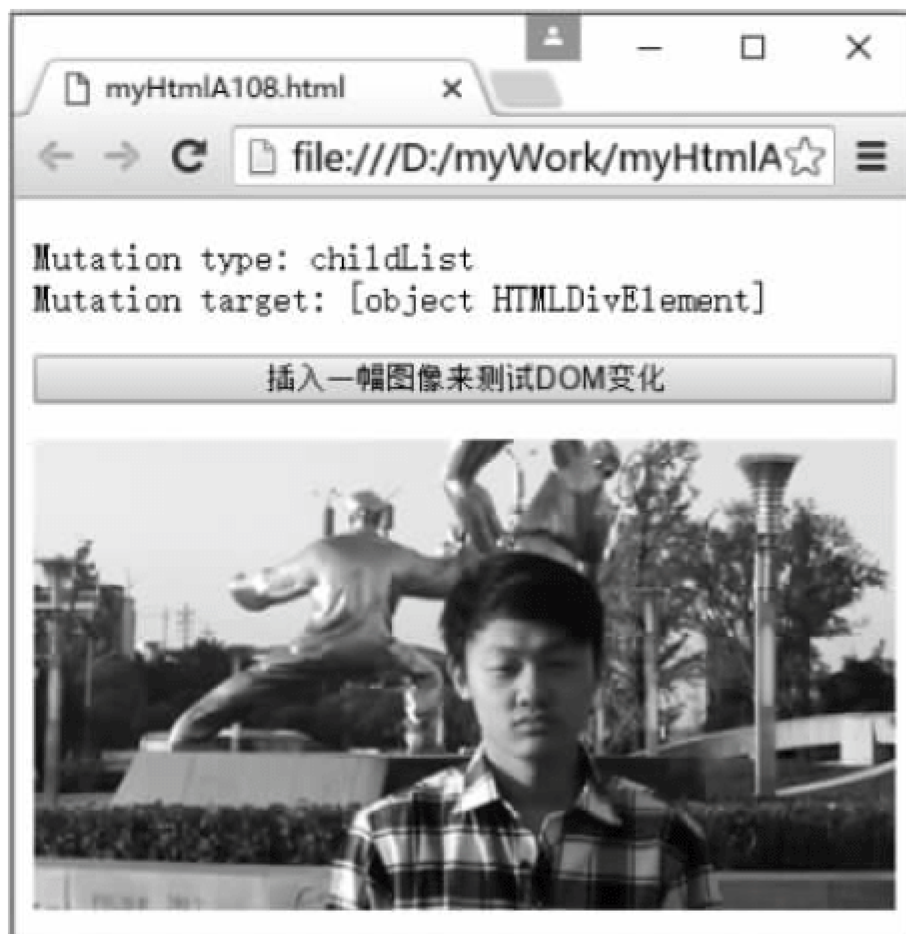


图　609-1

```
<!doctype html><html><head><meta charset="UTF-8"></head>
<body><p id="myInfo"></p>
<p><input type="button" value="插入一幅图像来测试 DOM 变化" onclick="changeDiv();" style="
width:375px"></p><div id="myDiv"></div>
<script>
 var onChange = function(myRecords) {
 myRecords.map(function(myRecord) {//输出检测到的变化信息
  document.getElementById('myInfo').innerText = 'Mutation type: '+
                 myRecord.type + '\n Mutation target: ' + myRecord.target;
 });};
 var myDiv = document.getElementById('myDiv');
 var myMutationObserver = new window.MutationObserver(onChange);
 var myOptions = {'childList': true,'subtree': true};
 myMutationObserver.observe(myDiv, myOptions);
 function changeDiv() {
  var mySpan = document.createElement("span");
```

```
    mySpan.innerHTML = "< img src = 'img/a108.jpg'>";
    myDiv.appendChild(mySpan); }
</script></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，var myMutationObserver＝ new window.MutationObserver(onChange)用于创建突变监视器，myMutationObserver.observe(myDiv，myOptions)用于设置监视对象和监视选项。当DOM对象树发生任何变动时MutationObserver就会得到通知，在概念上很接近事件。当DOM发生变动时会触发MutationObserver事件。MutationObserver与事件有一个本质的不同：事件是同步触发，也就是说DOM发生变动立刻会触发相应的事件；MutationObserver则是异步触发，DOM发生变动以后并不会马上触发，而是要等到当前所有DOM操作都结束后才触发。这样设计是为了应付DOM变动频繁的情况。举例来说，如果在文档中连续插入1000个段落(p元素)会连续触发1000个插入事件并执行每个事件的回调函数，这很可能造成浏览器的卡顿；而MutationObserver完全不同，它在1000个段落都插入结束后才会触发，而且只触发一次。MutationObserver所观察的DOM变动包含以下类型。

(1) childList：子元素的变动。

(2) attributes：属性的变动。

(3) characterData：结点内容或结点文本的变动。

(4) subtree：所有下属结点(包括子结点和子结点的子结点)的变动。

想要观察哪一种变动类型，就在option对象中指定它的值为true。需要注意的是，不能单独观察subtree变动，必须同时指定childList、attributes和characterData中的一种或多种。除了变动类型，option对象还可以设定以下属性。

(1) attributeOldValue：值为true或者为false。如果为true，则表示需要记录变动前的属性值。

(2) characterDataOldValue：值为true或者为false。如果为true，则表示需要记录变动前的数据值。

(3) attributesFilter：值为一个数组，表示需要观察的特定属性(例如['class'，'str'])。

DOM对象每次发生变化都会生成一条记录，这个记录对应一个MutationRecord对象，该对象包含了与变动相关的所有信息。MutationRecord对象包含了DOM的相关信息，有如下属性：

(1) type：观察的变动类型(attribute、characterData或者childList)。

(2) target：发生变动的DOM对象。

(3) addedNodes：新增的DOM对象。

(4) removeNodes：删除的DOM对象。

(5) previousSibling：前一个同级的DOM对象，如果没有则返回null。

(6) nextSibling：下一个同级的DOM对象，如果没有就返回null。

(7) attributeName：发生变动的属性。如果设置了attributeFilter，则只返回预先指定的属性。

(8) oldValue：变动前的值。这个属性只对attribute和characterData变动有效，如果发生childList变动，则返回null。

此实例的源文件名是myHtmlA108.html。

610　允许或禁止背景跟随浏览器的滚动条滚动

此实例主要通过设置background-attachment属性的不同属性值实现允许或禁止背景跟随浏览器的滚动条滚动。当在Google Chrome浏览器中显示该页面时，单击“允许背景随着滚动条滚动”按

钮后，拖动浏览器的滚动条滑块，则页面背景将跟随滚动条滚动；单击“禁止背景随着滚动条滚动”按钮后，拖动浏览器的滚动条滑块，则页面背景将不跟随滚动条滚动，如图 610-1 所示。有关此实例的主要代码如下。



图 610-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <script src="js/jquery-3.1.1.min.js"></script><script language="javascript">
  $(document).ready(function() {
   $("#myBtnScroll").click(function() {        //允许背景随着滚动条滚动
    $("html").css("background-attachment", "scroll");
   });
   $("#myBtnFixed").click(function() {         //禁止背景随着滚动条滚动
    $("html").css("background-attachment", "fixed");  });});
  </script>
  <style type="text/css">
    html { background-image: url('img/B206.jpg'); }
    div { width: 100%; height: 5000px;}
    button { width: 195px; }
    p { text-align: center;}
 </style></head>
<body><p><button id="myBtnScroll">允许背景随着滚动条滚动</button>
   <button id="myBtnFixed">禁止背景随着滚动条滚动</button></p>
<div></div></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，background-attachment 属性用于设置背景图像是否固定或者随着页面的其余部分滚动，该属性支持的属性值的说明如下。

(1) scroll：默认值，表示背景图像会随着页面中其余部分的滚动而移动。

(2) fixed：表示当页面的其余部分滚动时背景图像不会移动。

(3) inherit：规定应该从父元素继承 background-attachment 属性的设置。

此实例的源文件名是 myHtmlB206.html。

611　在百度地图中根据经度和纬度指示位置

此实例主要实现了根据某地区的经度和纬度信息通过使用百度地图 API 显示该地区在地图上的位置。当在 Google Chrome 浏览器中显示该页面时将根据重庆的经度和纬度信息(经度是 106.54、纬度是 29.59)显示重庆在地图上的位置，如图 611-1 所示。有关此实例的主要代码如下。



图 611-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
  body, html, #myMap { width: 100%; height: 100%;overflow: hidden; margin: 0;
font-family: "微软雅黑"; }
 </style>
 <script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=
8YlC3C8g5cZcG4I2pozDCuxFuPjY8FMy"></script>
</head>
<body><div id="myMap"></div>
<script type="text/javascript">
    var myMap = new BMap.Map("myMap");
    //重庆的经度是106.54、纬度是29.59
    var myPoint = new BMap.Point(106.54, 29.59);
    //12 表示缩放比例
    myMap.centerAndZoom(myPoint, 12);
    var myMarker = new BMap.Marker(myPoint);
    myMap.addOverlay(myMarker);
    myMap.panTo(myPoint);
//    //以下代码是根据 IP 定位当前位置
//    var myMap = new BMap.Map("myMap");
//    var myPoint = new BMap.Point(0, 0);
//    myMap.centerAndZoom(myPoint, 12);
//    var myGeolocation = new BMap.Geolocation();
//    myGeolocation.getCurrentPosition(function(r) {
//        if (this.getStatus() == BMAP_STATUS_SUCCESS) {
//            var mk = new BMap.Marker(r.point);
//            myMap.addOverlay(mk);
//            myMap.panTo(r.point);
//            alert('您的位置: '+r.point.lng+','+r.point.lat);
//        }
//        else {alert('failed'+this.getStatus());}
```

```
//      }, { enableHighAccuracy: true })
</script></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，src="http://api.map. baidu.com/api? v=2.0&ak=8YlC3C8g5cZcG4I2pozDCuxFuPjY8FMy"中的 ak 号码需要向百度地图开发者平台申请，在用于商业开发时百度可能会收取一定的费用，否则在运行时将给出如图 611-2 所示的提示信息。

此实例的源文件名是 myHtmlA114.html。



图 611-2

612 在关闭页面前弹出消息框提示用户确认

此实例主要设置页面 body 的 onbeforeunload()方法的返回值，从而实现在关闭页面前弹出消息框提示用户做最后的关闭确认。当在 Google Chrome 浏览器中显示该页面时，如果使用鼠标单击浏览器右上角的“关闭(×)”按钮，则将弹出一个“确认导航”消息框，提示“请慎重考虑一下：确定要离开此页吗?”，如图 612-1 所示；单击“离开此页”按钮，则关闭页面，单击“留在此页”按钮，则返回页面。有关此实例的主要代码如下。



图 612-1

```
<!doctype html><html><head><meta charset="UTF-8">
 <style type="text/css">
   img { width: 400px; height: 250px; border-radius: 10px; }
 </style></head>
<body onbeforeunload="return '请慎重考虑一下: ';">
<img src="img/B316.jpg"></body></html>
```

上面有底纹的代码是此实例的核心代码。在该部分代码中，<body onbeforeunload="return '请慎重考虑一下：';">用于在关闭页面前给用户一个提示信息。

此实例的源文件名是 myHtmlB316.html。

图书资源支持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的资源，有需求的读者请扫描下方的“书圈”微信公众号二维码，在图书专区下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

我们的联系方式：

地　　址：北京海淀区双清路学研大厦 A 座 707

邮　　编：100084

电　　话：010－62770175－4604

资源下载：http://www.tup.com.cn

电子邮件：weijj@tup.tsinghua.edu.cn

QQ：883604（请写明您的单位和姓名）



书圈

用微信扫一扫右边的二维码，即可关注清华大学出版社公众号“书圈”。